(12)

Carderock Division
Naval Surface Warfare Center
Bethesda, MD 20084-5000

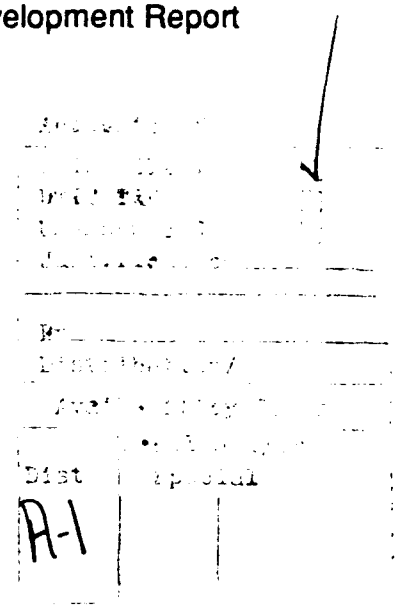
PAS-92-27 June 1992
Propulsion and Auxiliary Systems Department
Research and Development Report

MAGNETIC FLUX - LOAD CURRENT INTERACTIONS IN FERROUS CONDUCTORS

by
Michael J. Cannell
Richard A. McConnell

Carderock Division
Naval Surface Warfare Center
Bethesda, MD 20084-5000

PAS-92-27 June 1992
Propulsion and Auxiliary Systems Department
Research and Development Report



MAGNETIC FLUX - LOAD CURRENT INTERACTIONS IN FERROUS CONDUCTORS

by
Michael J. Cannell
Richard A. McConnell

CONTENTS

TABLES

FIGURES

FIGURES (Continued)

PAGE

ABSTRACT

A modeling technique has been developed to account for interactions between load current and magnetic flux in an iron conductor. Such a conductor would be used in the active region of a normally conducting homopolar machine. This approach has been experimentally verified and its application to a real machine demonstrated. Additionally, measurements of the resistivity of steel under the combined effects of magnetic field and current have been conducted.

ADMINISTRATIVE INFORMATION

The magnetic analysis effort described herein was initiated under the Independent Research and Independent Exploratory Development Program at this center under Work Unit 1-2711-100 and subsequently supported by the Office of Naval Technology under Program Element 62121N Projects RH21E41 and RH21E42. The work was performed in the Electrical Propulsion and Machinery Systems Branch of the Propulsion and Auxiliary Systems Department at the Annapolis Detachment, Carderock Division, Naval Surface Warfare Center.

ACKNOWLEDGMENTS

The authors wish to acknowledge the assistance of the following individuals for the construction and setup of the experimental hardware: Mr. John Stevens, Mr. Harold Surosky, Mr. Edward Lee, Mr. Patrick Owens, Mr. Patrick Reilly, Mr. Stanley Kazmerski, and Mr. Kenneth Sabel. Mr. Robert Lari of Vector Fields Inc. provided valuable assistance with the finite element modeling.

INTRODUCTION

BACKGROUND

Homopolar machines produce torque by the direct interaction of current, in a conductor, with a separately produced magnetic flux in the active region of the machine . The two primary functions of the active region are the transport of the load current and the magnetic flux at right angles to each other. In a homopolar machine with

superconductive excitation, the conductor of choice to conduct the load current through the active region of the machine is copper. Copper has good electrical properties but has a magnetic permeability the same as air. The low permeability is not a problem since most of the flux path is already air and the superconductive magnet provides the required ampere turns in a compact coil with low power requirements.

However, in a homopolar machine with normal excitation, a complete ferrous path must be provided because of the limited ampere turns available from a normal magnet. In this case, transporting the current through the active region of the machine becomes more difficult and involves more tradeoffs. The use of continuous copper sections provides an excellent current path but introduces a large effective air gap in the magnetic circuit. This results in significant increases in the size and power consumption of the field coils with direct negative impacts on the efficiency and size of the machine. One alternative is the use of a slotted iron rotor with copper bars in the slots to carry current and the ferrous material remaining between the slots to carry the magnetic flux. The primary disadvantage of this approach is that the active region must be enlarged, since only a portion of it is available for each of its two functions. A second alternative is the use of iron conductors to carry both the load current and the magnetic flux. The advantage of this approach is that the entire volume of the active region is available for both flux and current transport since the conductors are ferrous. Because of this combination of functions the active region is smaller, thereby reducing the overall machine size. This is the approach that was selected for the demonstration contra rotating motor being developed as part of this task.

There are, however, interactions between the load current and the transverse magnetic flux that must be analyzed and accounted for in the design of the machine. The load currents in the iron bars generate an internal magnetic field intensity that must be added vectorialy to the torque producing radial field from the central field coil. This effect is shown in Fig. 1 where ϕ_a is flux from the load current passing axially through

the iron bars, and ϕ_r is the radial flux from the field coil. In superconducting machines which use copper bars, ϕ_a is very small because the permeability of copper is very low. However, in a normally conducting machine with iron bars, ϕ_a is significant and the resulting combination of the two fluxes (shown as part c of Fig. 1) produces a higher total magnetic field in the bar. All magnetic materials are non-linear and most have a lower permeability at higher magnetic field intensities. Thus the working flux in the machine sees a lower effective permeability in the iron bars due to presence of the axial current, which must be accounted for by increasing the amp-turns of the field coil.

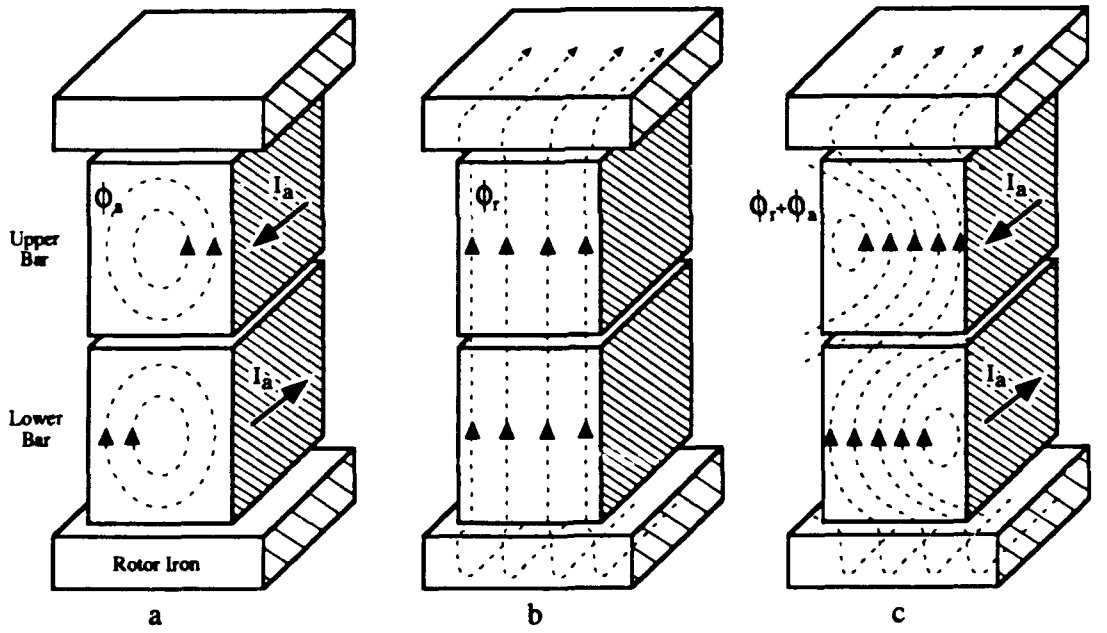


Fig. 1. Components of Load Current Flux Interaction.

a) Circular flux from armature current in upper and lower iron bars.
b) Radial flux from the field winding with no armature current.
c) Interaction of field and armature flux when both field and bar currents are present.

A second effect which may need to be accounted for is the increase in resistivity of the iron conductors due to magneto resistive effects. That is, in the presence of high magnetic fields, some irons tend to show an increase in resistivity. In most machines this is not a problem because iron is not normally used to carry current. However in the normal conducting homopolar motor, an increase in resistivity of the current carrying iron bars will increase heat production and lower the efficiency of the machine. This effect is probably small and was not modeled. Rather, iron bars of the same type as will be used in the demo motor were experimentally tested and the magnetic field was shown to have little effect.

APPROACH

The approach taken for the flux - load current interaction portion of this work was to develop analytical methods that could be applied to homopolar machine design in general rather than an approach that addressed the problem only for the particular geometry of the motor under development. The basic method we selected to account for the flux - load current interaction was a combination of two 2 dimensional finite element models. The flux-load interaction's influence on the flux level in the machines is actually a 3 dimensional effect. However, 3 dimensional models are large and cumbersome, and each model would be tied to the specific geometry of the bars. Instead, this technique consisted of using a 2 dimensional finite element program to model the flux-load interaction's effect on the permeability of the iron bars. These modified BH curves were then used to represent the bar iron in another 2 dimensional model which calculated the total working flux and torque in the machine.

The capability of finite element techniques to accurately model the flux load interaction was first verified experimentally with an apparatus that could be modeled with our existing 2D finite element software (PE2D by Vector Fields). The geometry selected consisted of a nominal 1 in. x 3 in. steel bar placed between the poles of a water cooled electromagnet which established a transverse magnetic induction. At the same time a

current was established in the longitudinal direction in the bar. The current in the magnet was linearly ramped up while the magnet current and the average transverse magnet induction were recorded on an X Y plotter. This was done for bar currents of 0, 2,000, 4,000, 6,000, 8,000, and 10,000 amps and compared to the results of a finite element model of the apparatus.

Once the finite element method was verified using a single bar, the complex flux interaction of multiple bars side by side, shown in Fig. 2, was modeled using finite element techniques. This finite element model is more complicated because the boundary conditions around any given bar (symmetry plus a constant) do not fit the typical boundary conditions allowed for in most finite element packages. An in-house finite element program was written to allow for this boundary condition. The in-house program was verified by running iron bar models using regular symmetry conditions and comparing the results with the PE2D finite element package used to model the experimental test setup. After verification, the in-house program was then run using the symmetry plus a constant boundary condition for various states of radial flux and axial current to create effective BH characteristics. These BH curves were then used for the iron bar material in the 2D finite element model of the entire machine.

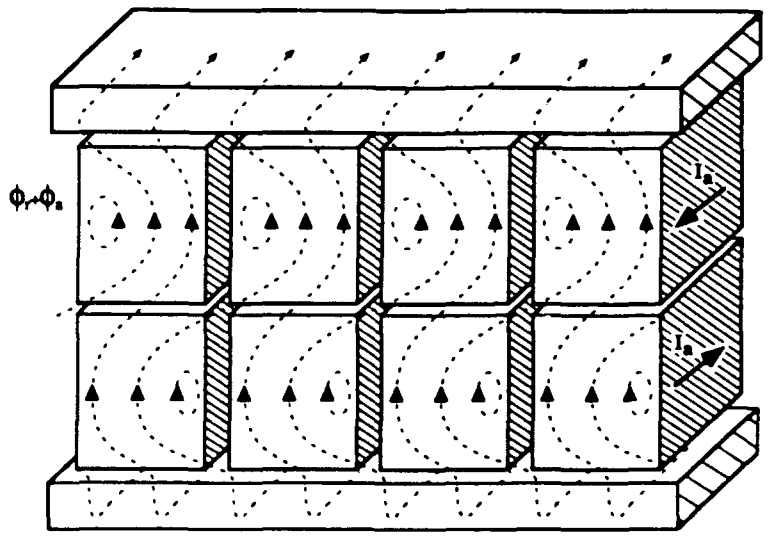


Fig. 2. Several Bars Side by Side to Show Complex Flux Interaction and Boundary Condition.

The approach for the magneto resistive effect was experimental in nature. The resistance of the steel bar was measured under realistic magnetic induction and current conditions to determine if the resistance changed in response to the transverse magnetic induction.

EXPERIMENTAL DETERMINATION OF FLUX-CURRENT INTERACTION

TEST SETUP

The basic test setup consisted of a ferrous bar with a cross section of .972 in. x 2.883 in. This bar was fitted with a trapezoidal adapter bar on each edge to provide a gradual transition to the magnet pole face. Four G-10 (glass-epoxy) dowels maintained alignment between the test bar and each adapter. A 0.045 in. air gap was maintained between the test bar and each adapter by plastic shim washers on the dowel pins. This gap allowed insertion of a Hall effect probe to measure the transverse magnetic induction. Figure 3 shows the design of the test bar and adapters. The test bar was fitted with thermocouples, voltage taps, and field pickup coils. This assembly was inserted between the poles of a 15 in. water cooled electromagnet and connected to a pair of 5,000 amp power supplies with water cooled power cables. The instrumented test bar is shown in place between the poles of the magnet in Fig. 4. The electromagnet was connected to a programmable power supply controlled by an arbitrary wave form generator. The instrumentation set up is shown in Fig. 5.

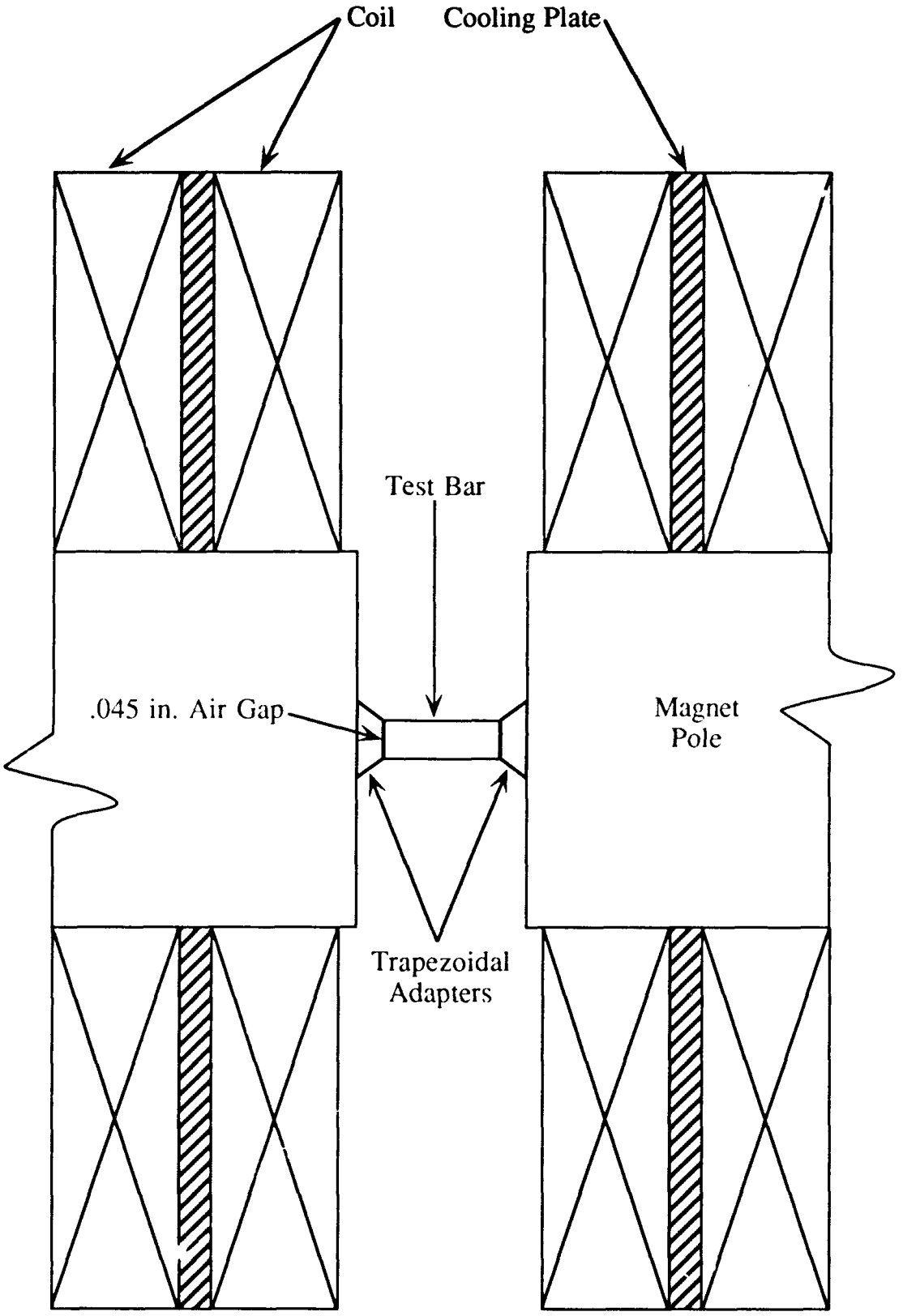


Fig. 3. Test Bar and Adapters Installed Between Magnet Poles

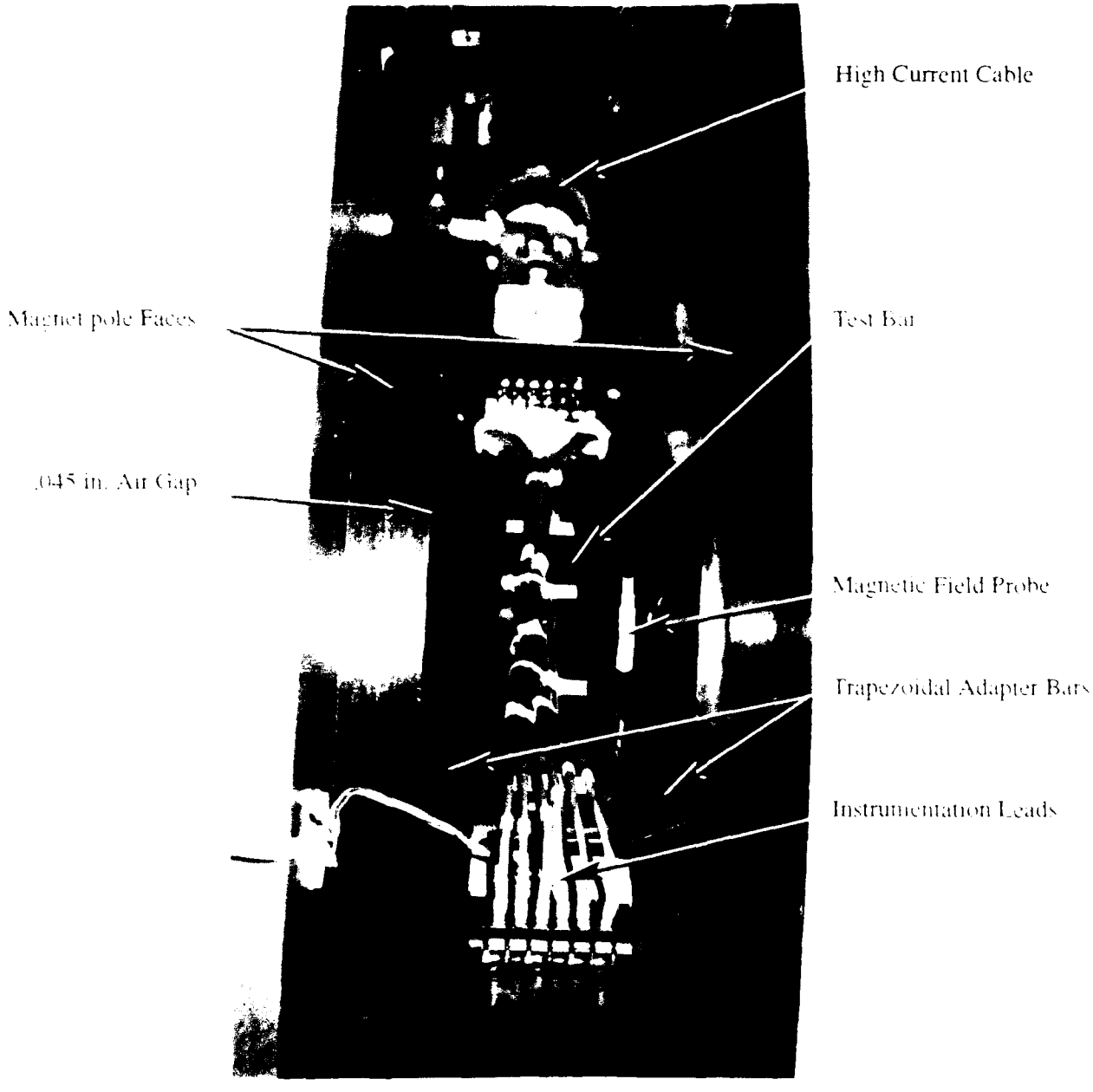


Fig. 4. Instrumented Test Bar Installed Between the Magnet Poles.

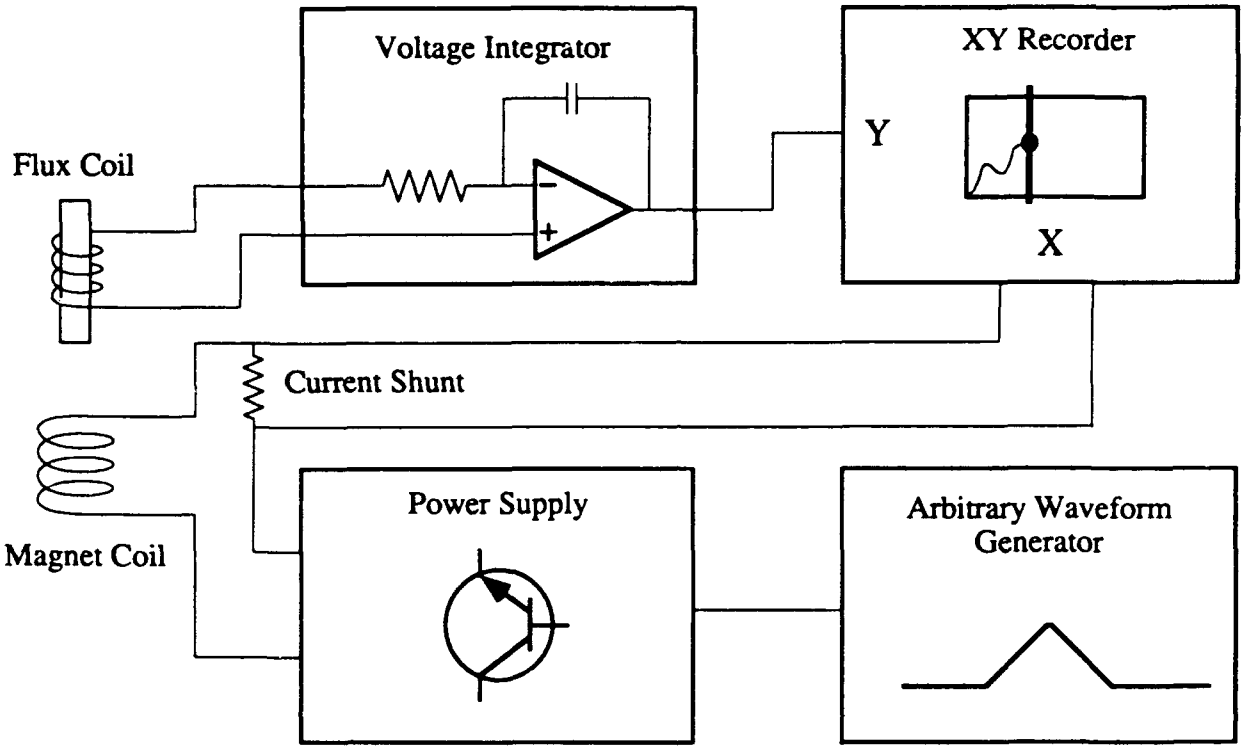


Fig. 5. Instrumentation Setup

The average value of the transverse magnetic induction was obtained by integrating the output voltage from the pickup coils with a low drift integrator. The voltage induced in the field pickup coil by the changing magnetic flux is given by

$$V = N\frac{d\Phi}{dt} . \tag{1}$$

Where Φ is the transverse magnetic flux and N is the number of turns in the pickup coil. This voltage is applied to the input of a voltage integrator whose output voltage is given by

$$V_{out} = \frac{1}{RC}\int_0^t V dt . \tag{2}$$

Substituting Eq 1 into Eq 2 results in

$$V_{out} = \frac{1}{RC}\int_0^t N\frac{d\Phi}{dt}dt \tag{3}$$

which when integrated gives

$$V_{out} = \frac{N}{RC}(\phi(t)-\phi(0)) \tag{4}$$

The integrator output is initially set to 0 which results in an output voltage of

$$V_{out} = \frac{N\Phi(t)}{RC} \tag{5}$$

where $\Phi(t)$ is the change in flux since the integrator output was set to zero.

The relationship between magnetic induction $\vec{B}$ and flux through a surface S is given by

$$\Phi = \int_S \vec{B}\cdot\vec{n}da . \tag{6}$$

With the geometry of this setup, $\vec{B}$ is normal to S, and since we are interested in the average value of $\vec{B}$, Eq 6 becomes

$$\Phi = B_{avg}LW \quad . \tag{7}$$

Where B_{avg} is the average normal magnetic induction and L and W are the length and width of the surface S. Combining Eqs 5 and 7 and solving for B_{avg} we have

$$B_{avg} = \frac{V_{out}RC}{NLW} \quad . \tag{8}$$

Values of R and C were chosen such that

$$B_{avg} = \frac{V_{out}}{2} \quad . \tag{9}$$

Temperatures from the six type T thermocouples on the bar were directly monitored and recorded by a data acquisition system. The magnet current was monitored with an in line shunt and recorded on a X Y plotter. The bar current was monitored with in line shunts in the power supplies and also recorded by the data acquisition system. The voltage drop in the center 15 inches of the bar was monitored with 6 sets of voltage taps, 3 sets on each side. These were measured and averaged by the data acquisition system.

During initial checkout and calibration, a problem with the setup was encountered. If the magnet current was ramped up very slowly the curve of integrator output verses magnet current appeared proper and followed the same path both during ramp up and ramp down. If the ramp rate was increased, the two curves diverged from each other resulting in an open loop. The ramp up curve was below the slow ramp curve and the ramp down curve was above it, as illustrated in Fig. 6. The faster the ramp rate the more open the curves became. We considered using a slow ramp rate but encountered two secondary problems. The first was drift of the integrator output voltage due to input offset voltage. The second was heating of the uncooled test bar, particularly at the higher currents. The basic problem with the faster ramp had to be solved to insure accurate data.

The most likely cause appeared to be eddy currents in the copper cooling plates or in the iron yoke of the magnet assembly. The copper cooling plates were the most likely

source of the problem since they appear as low resistance shorted turns around the magnet pole and link 100 % of the magnet flux. The effective ampere turns in the magnet

is equal to the algebraic sum of the ampere turns in the coils plus the ampere turns due to eddy currents. During ramp up the eddy currents reduce the effective ampere turns and during ramp down they increase the effective ampere turns. This explanation is consistent with our observations.

We decided to use a fast ramp rate and to compensate for the eddy currents in the magnet assembly. We know that the induced voltages causing the eddy currents are given by

$$V = \frac{d\Phi}{dt} . \tag{10}$$

The eddy currents, I, are equal to

$$I = \frac{V}{R} \tag{11}$$

where V is the induced voltage and R is the effective resistance of the eddy current path or paths. If Eq 10 is substituted in Eq 11 we have

$$I = \frac{d\Phi}{dt}\,\frac{1}{R} . \tag{12}$$

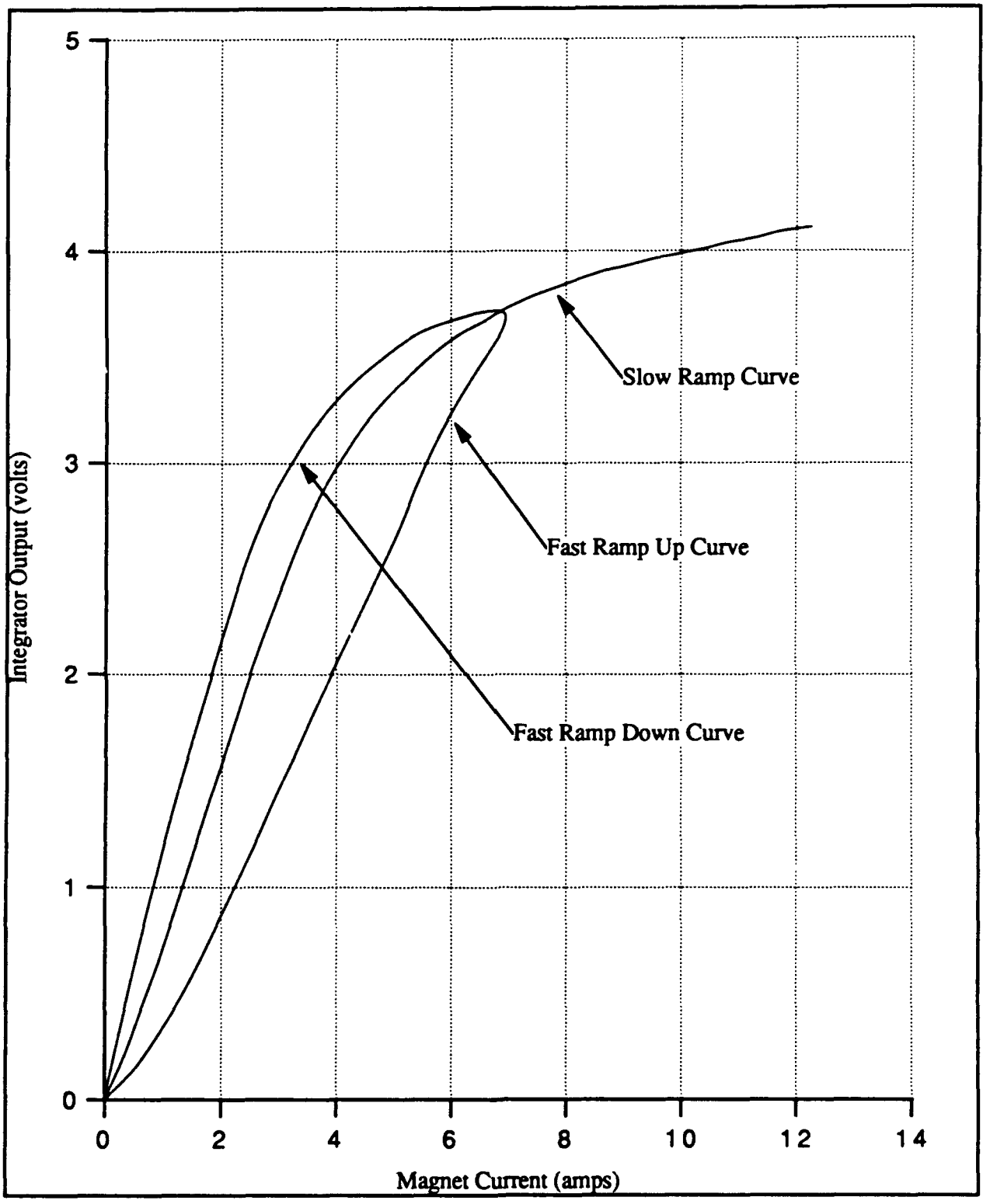


Fig. 6. Fast and Slow Ramp Rate Curves of Integrator Output Versus Magnet Current

In this equation $d\Phi/dt$ is the only parameter that changes with ramp rate, R is a function of the geometry and material properties but would be difficult to determine accurately. Since $d\Phi/dt$ can be directly monitored with a coil that links the flux we can experimentally compensate for the eddy currents without actually determining their magnitudes. A three (3) turn coil was placed around the electromagnet pole and its output connected to a 10 turn potentiometer used as a voltage divider. The output from the voltage divider was connected in series with the output from the magnet current shunt and the series combination was connected to the X Y plotter. Thus a voltage term proportional to $d\Phi/dt$ was subtracted from the current signal. By trial and error a potentiometer setting was found that resulted in good closed curves of flux verses magnet current. These curves did not change with ramp rate and they matched the curves obtained at a slow ramp rate. This setting was used during all subsequent measurements. This corrected magnet current is referred to as the effective magnet current.

Magnetic measurements are normally made on demagnetized samples to eliminate the effect of residual magnetization on the results. In this setup it was not practical to demagnetize the sample; therefore a compensating method was developed. This consisted of making a first set of measurements with the magnetic induction in the same direction as the residual magnetization in the test bar and then reversing the input to the magnet and repeating the measurements with the magnetic induction in the opposite direction. These two sets of measurements were averaged, as shown in Fig. 7, giving the same results as if the sample had been demagnetized prior to each set of measurements.

PROCEDURE

The axial current in the test bar was first set to the desired value with the power supplies in the current regulating mode to prevent drift. The integrator output was set to zero and the current in the magnet was then ramped up to a predetermined level and

returned to zero, at a constant rate, by means of the wave form generator programmed with a triangular wave. The outputs from the integrator and the magnet current shunt were recorded on a XY potter as the current was ramped up and then down. The magnet leads were then reversed and the above procedures repeated exactly with the results recorded on the same graph in the XY plotter.

The test bar was allowed to cool and the measurements were repeated for each of the bar currents. The curves were then digitized to allow averaging of the two curves for each test condition. The vertical axis was scaled to indicate B_{avg}, the average transverse magnetic induction.

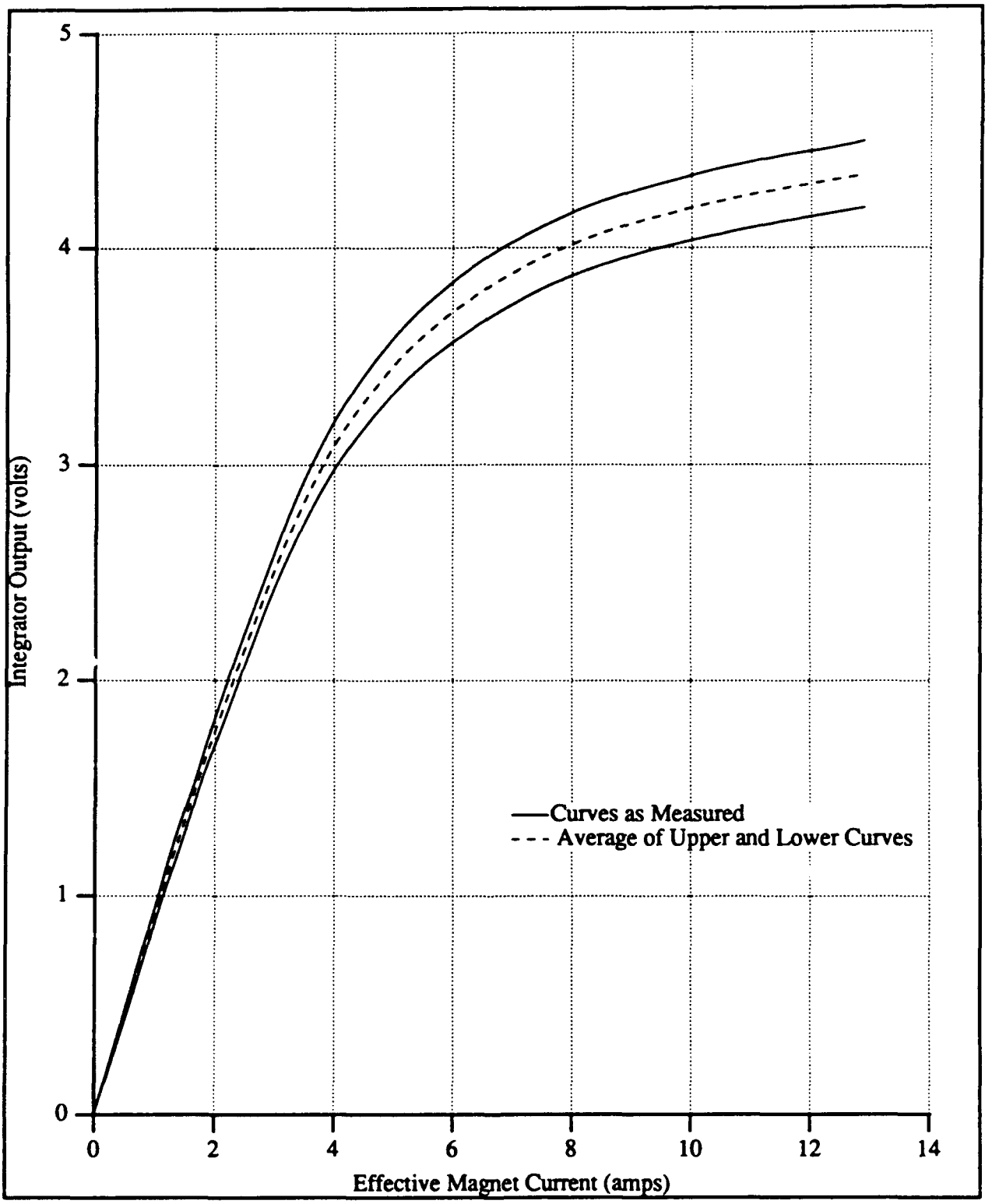


Fig. 7. Typical Curves of Integrator Output Versus Magnet Current and Average of the Two Curves

RESULTS

The results for each of the bar test currents are shown in Figs. 7 through 13. The experimental curves shown are the averages of the two sets of measurements as previously discussed. The discrete data points shown are the results from the finite element model of the experiment. As expected, the axial current in the bar had a significant effect on the magnet current required to produce a given average transverse magnetic induction. It is interesting to note the trend for all the curves to converge at high values of magnet current, as shown in Fig. 14, which corresponds to high values of magnetic field intensity. At high values of magnetic field intensity the steel is at or near magnetic saturation and the addition of the bar current has little effect. It should also be noted that despite the similarity, these curves are not the same as normal magnetization curves for steel. The two air gaps and the iron in the electromagnet which are included in the path have a significant impact on the results. In addition, the effect of the load current on the effective magnetic properties is dependent on the geometry of the bar. Thus the results do not represent material properties as such, but are intended to verify the finite element modeling approach as discussed in the next section.

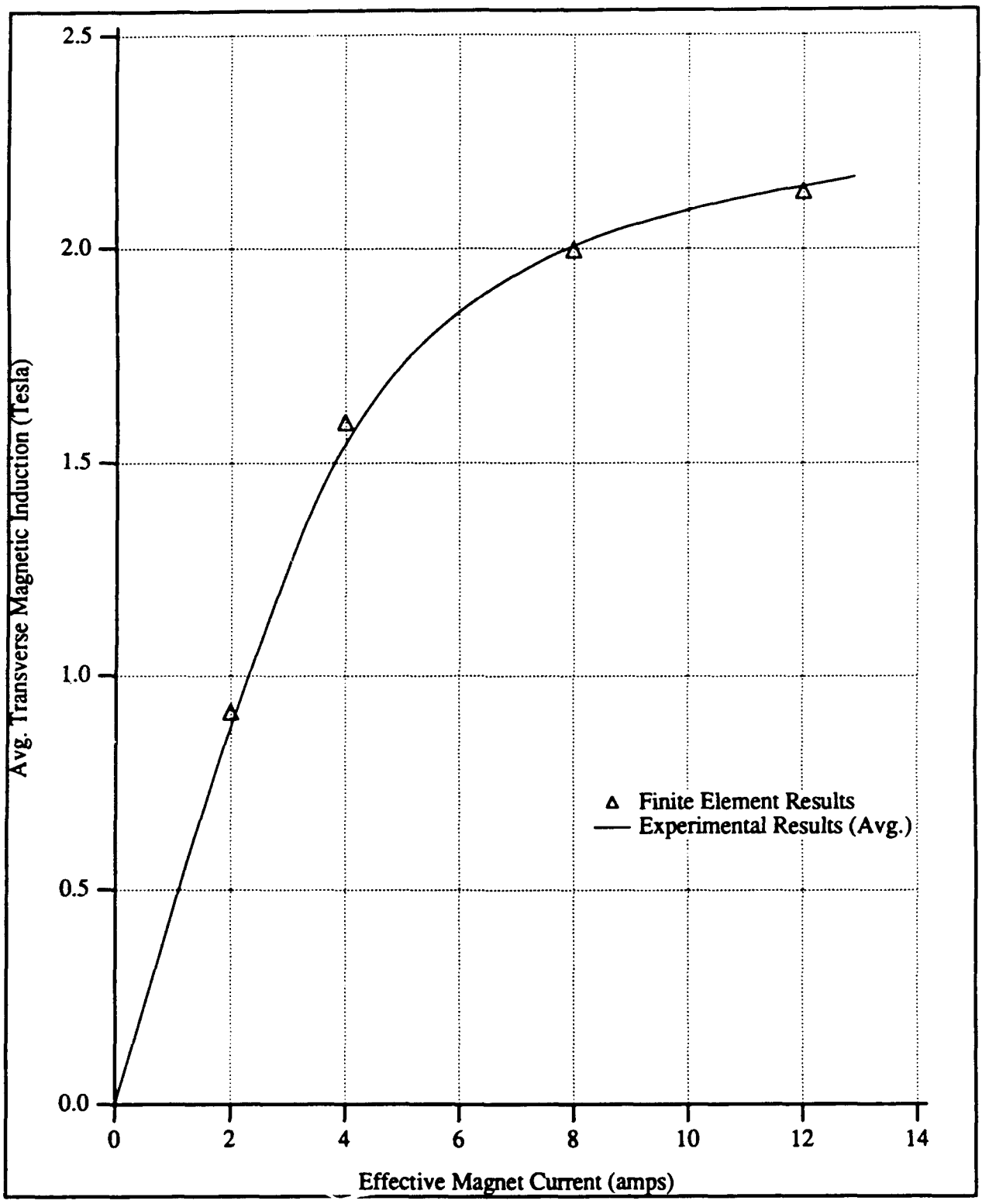


Fig. 8. Magnetic Induction Versus Magnet Current
at 0 Amps Bar Current

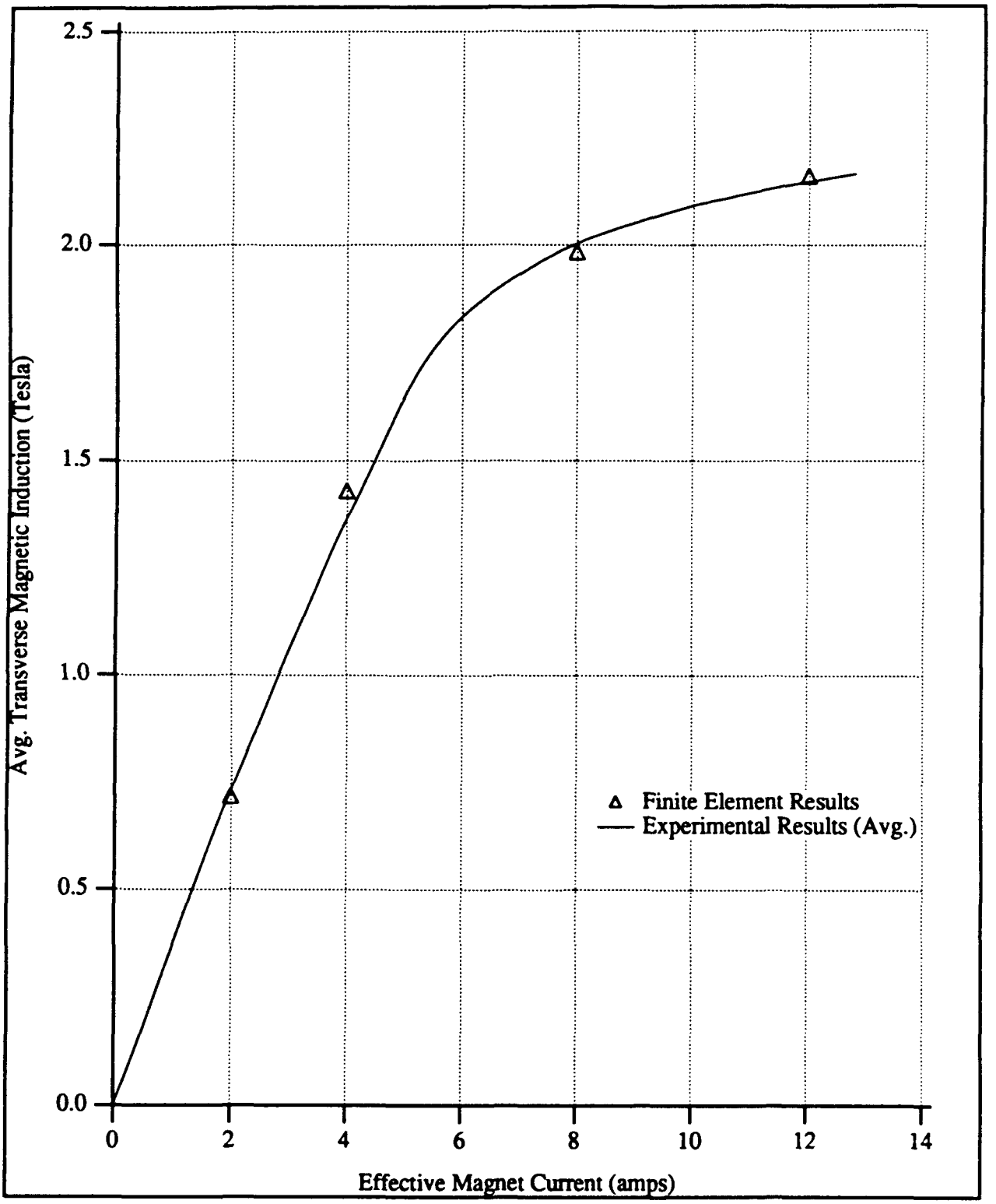


Fig. 9. Magnetic Induction Versus Magnet Current
at 2,000 Amps Bar Current

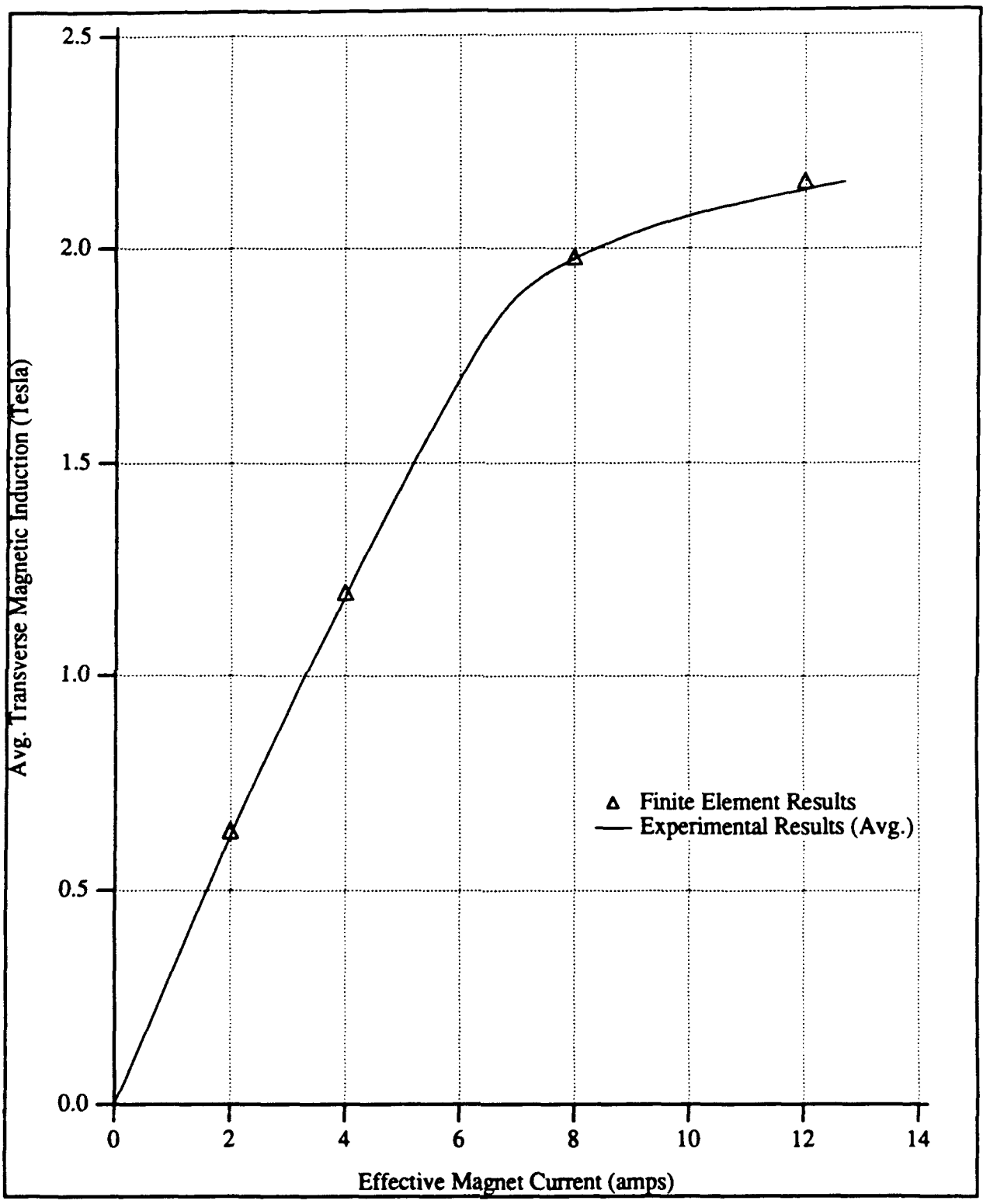


Fig. 10. Magnetic Induction Versus Magnet Current
at 4,000 Amps Bar Current

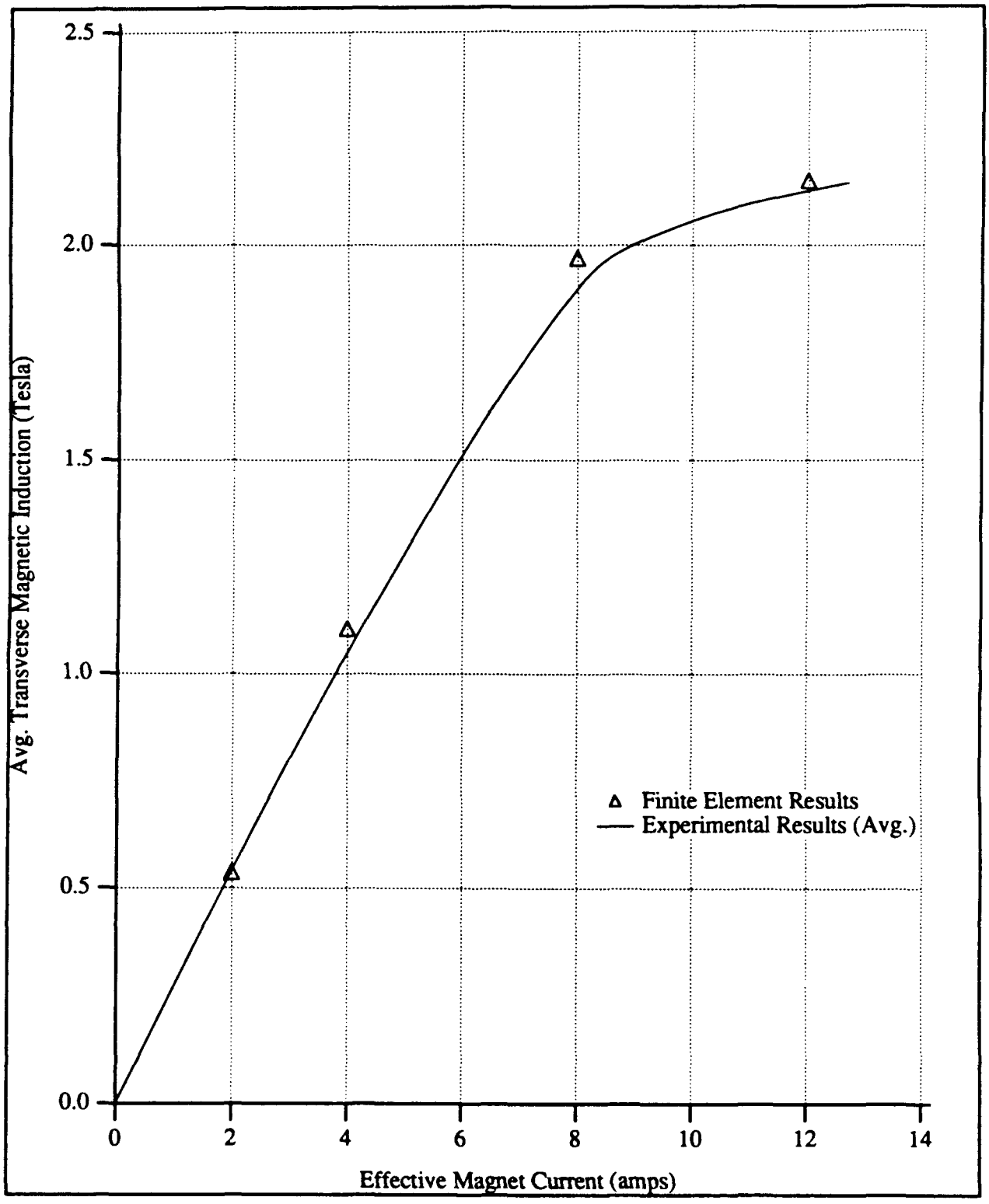


Fig. 11. Magnetic Induction Versus Magnet Current
at 6,000 Amps Bar Current

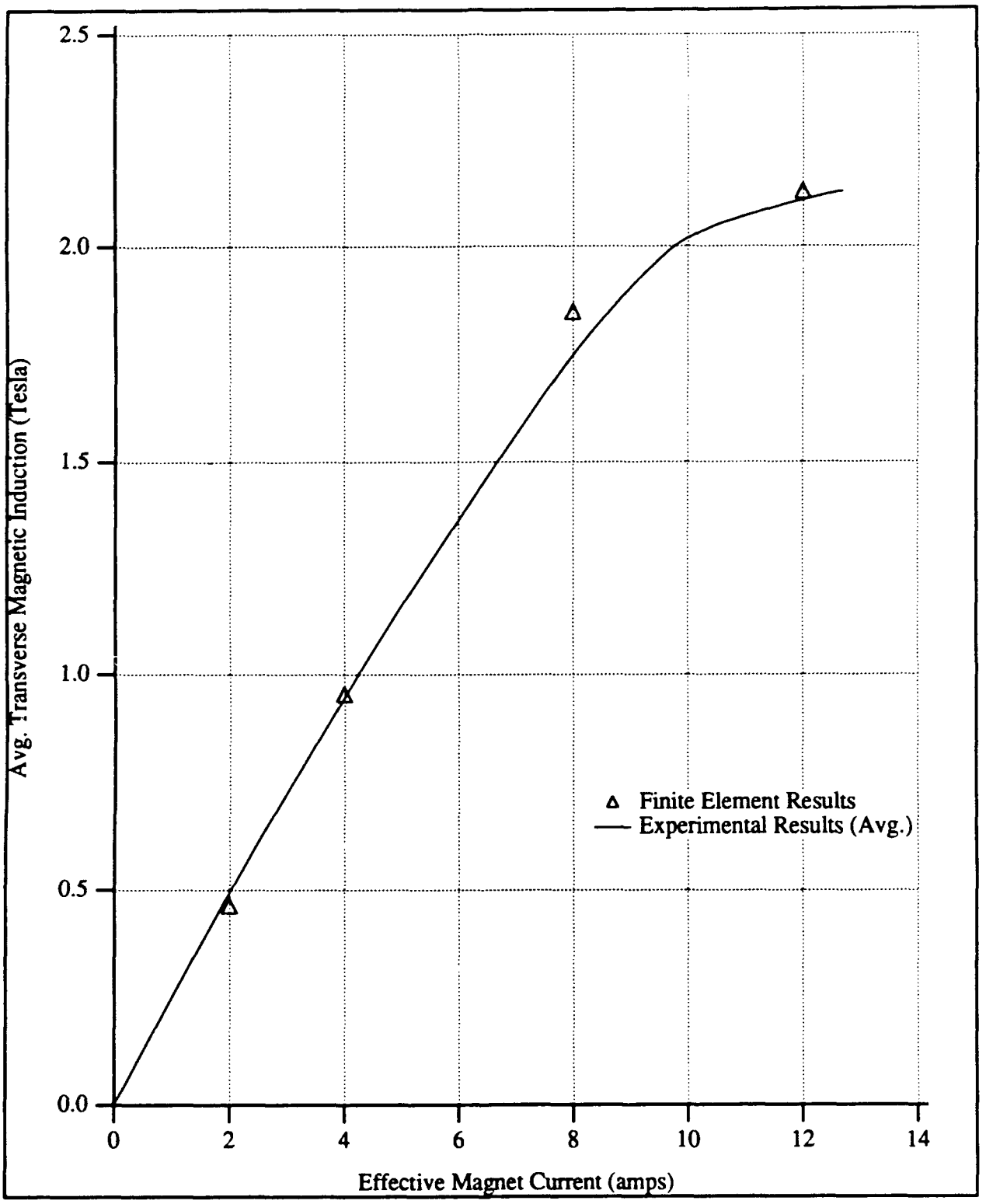


Fig. 12. Magnetic Induction Versus Magnet Current
at 8,000 Amps Bar Current

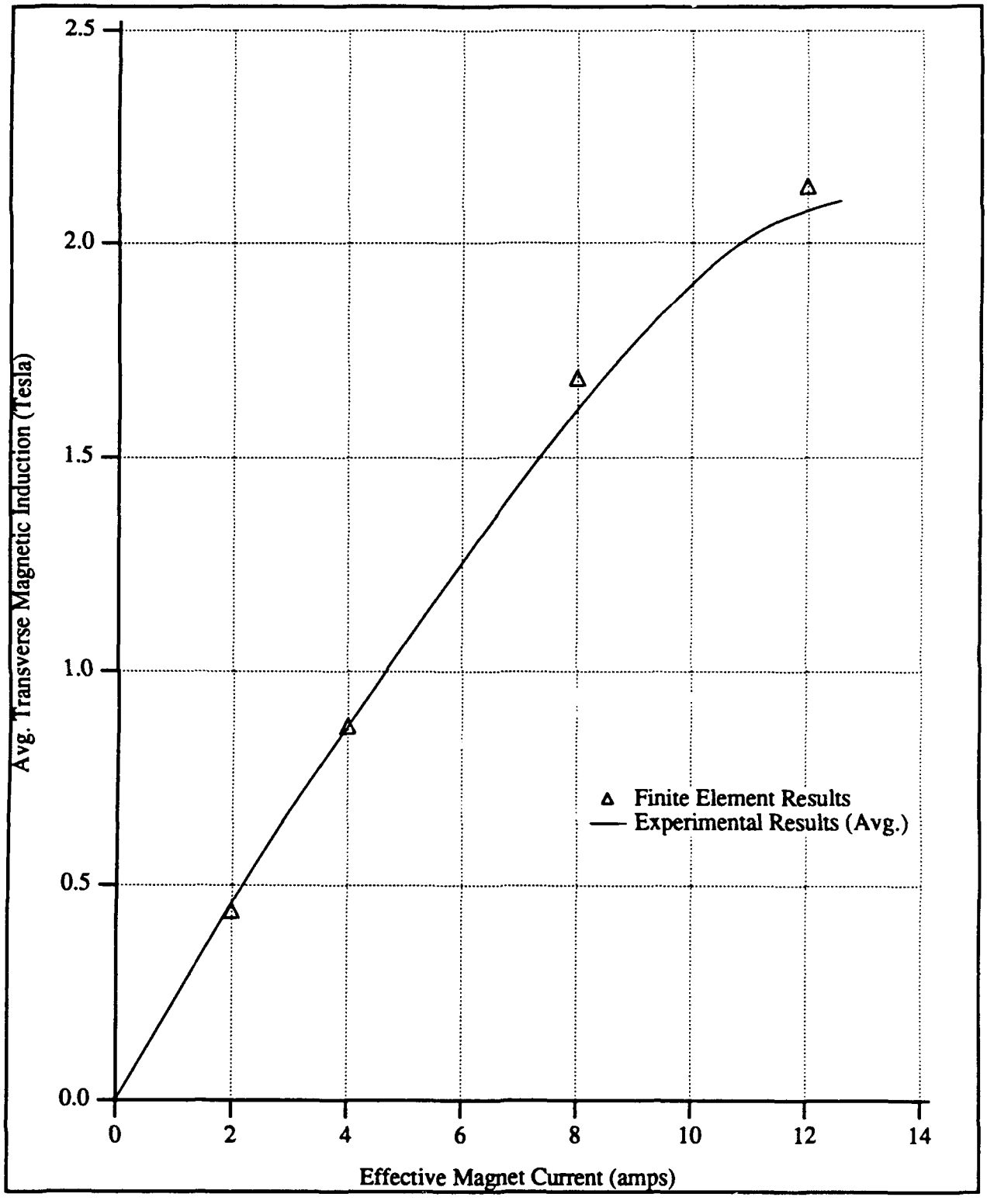


Fig. 13. Magnetic Induction Versus Magnet Current
at 10,000 Amps Bar Current

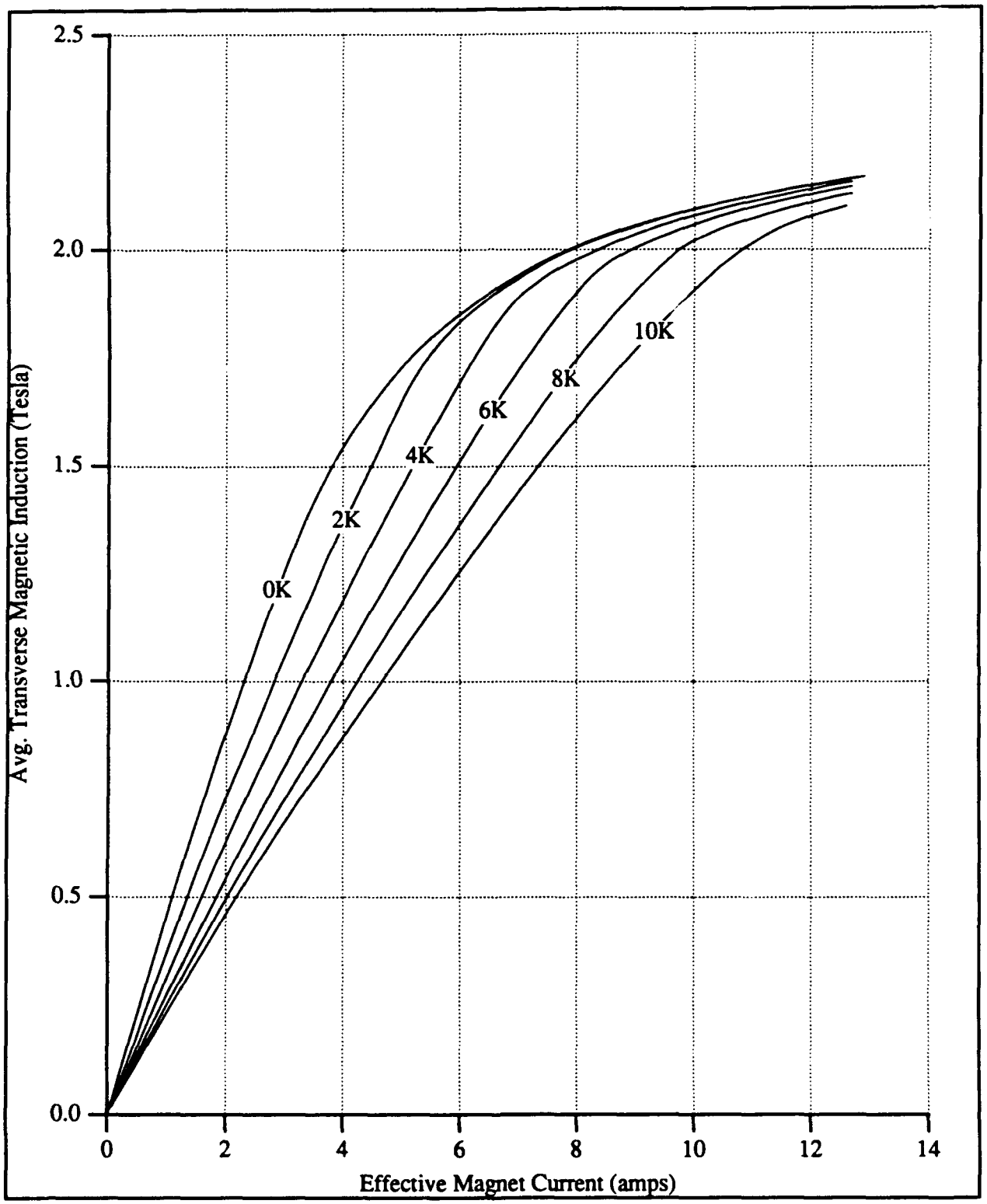


Fig. 14. Magnetic Induction Versus Magnet Current at 0, 2, 4, 6, 8, and 10 Kamps Bar Current

VERIFYING FINITE ELEMENT METHOD

As is evident from the curves shown in Figs. 8 through 13, the finite element model results correlate very well with the experimental data. For all cases of bar currents less than 6 K amps or field currents less than 6 amps, the variations between the measured and calculated values are within experimental error and are not significant. The error in the cases where the bar currents were more than 6 K amps and field currents were more than 6 amps is probably a result of the fact that the iron BH curve used in the finite element models was not an exact match for the iron actually used in the tests. Overall, this experiment verifies that the finite element approach that we developed accurately models the experimental setup and can be used for the model motor under development. This finite element approach is versatile and can be used for other machines that use ferrous conductors in the active region.

EXPERIMENTAL DETERMINATION OF MAGNETO RESISTIVE EFFECTS

TEST SETUP

The basic test setup was the same as previously described with the exception that some of the measurements were done in a different manner. The magnetic induction was directly measured by means of a flat Hall effect probe inserted in the gap between the test bar and the trapezoidal adapter, and all other parameters were recorded by a data acquisition system.

PROCEDURE

With the test bar in position between the poles of the electromagnet, a longitudinal current of 2,000 amps was established in the bar. The voltage drop from the voltage taps and the temperatures were recorded by the data acquisition system as the bar was allowed to heat to approximately 170 °F. The bar current was reduced to zero and the bar was cooled to room temperature. A transverse magnetic induction of 0.5 Tesla

was then established by the electromagnet. The longitudinal current of 2,000 amps was reestablished and the voltage and temperature measurements were again recorded as the bar heated. This procedure was repeated for magnetic inductions of 1.0, 1.5, and 2.0 Tesla at both 2,000 and 4,000 ampere longitudinal currents. Voltage drops were extracted from each of the data sets for temperatures of 100, 120, 140, and 160 °F. The six voltage drops for each test condition were averaged and the bar resistance calculated.

RESULTS

The test results for 2,000 amps are shown in Fig. 15 and for 4,000 amps in Fig. 16. As is evident from the curves, there is no significant variation in resistance of the steel bar as a result of the transverse magnetic induction. The maximum variation between the 2,000 and 4,000 amp data is 0.7 % and the maximum change with magnetic induction is 0.4 %. Both of these are well within experimental error for this setup. These variations are not significant in the design of a machine and do not need to be accounted for.

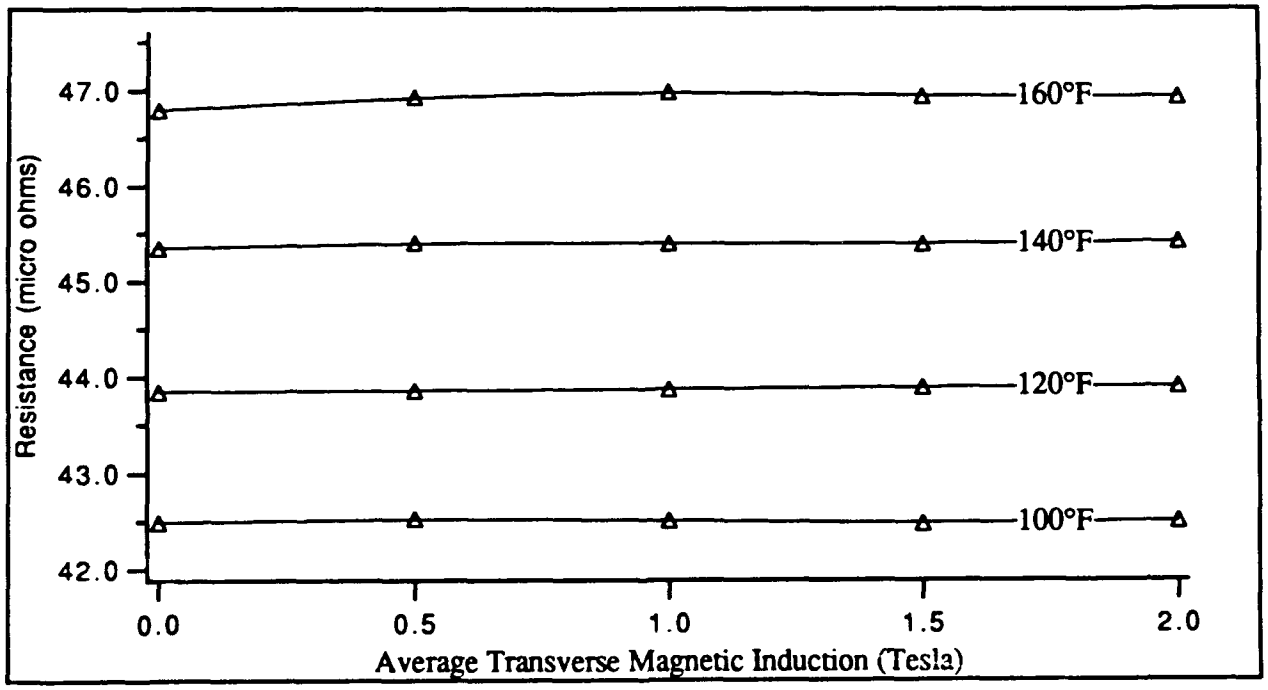


Fig. 15. Resistance Versus Magnetic Induction at 2,000 Amperes.

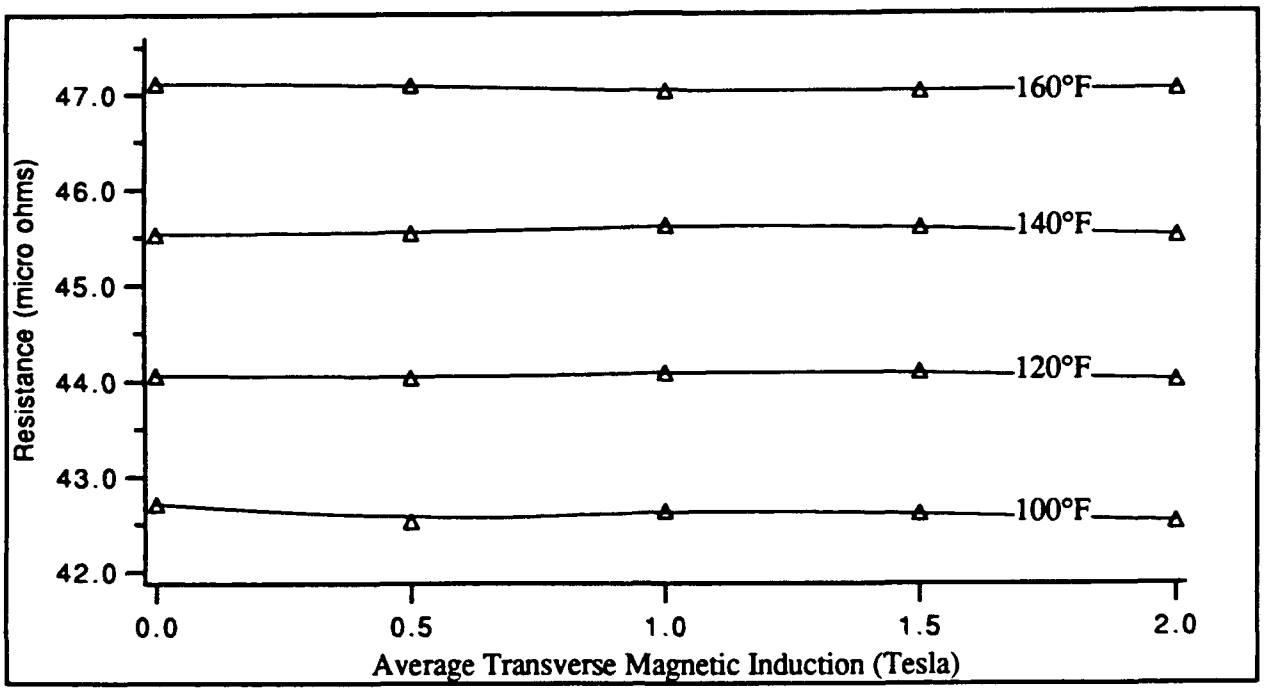


Fig. 16. Resistance Versus Magnetic Induction at 4,000 Amperes.

FINITE ELEMENT ANALYSIS OF FLUX LOAD INTERACTION

PROCEDURE

The complex flux interaction of multiple bars side by side (shown Fig. 2) was modeled using finite element techniques. This finite element model is more complicated because the boundary conditions around any given bar do not fit the typical boundary conditions allowed for in most finite element packages. A close examination of Fig. 2 shows that there is a symmetry condition between each bar. That is, the potential function used to describe one bar can describe any bar because the flux patterns are the same. The potential function for each bar is, however, *not* numerically equal. The uniform radial field implies that there must be a potential gradient circumferentially through the airgap so each bar will be at a higher potential level than the bar next to it. The complete radial field is three dimensional, therefore, it is impossible to model all the bars in the machine at once in only two dimensions because the radial field would apear from a point (not mathematically possible). However, a single upper and lower pair can

be modeled in two dimensions. It can be seen by the symmetry of Fig. 2 that an appropriate boundary condition for any pair of bars (upper and lower) is that for any given height, the magnetic potential at the center of the air space to the left of the bars is constrained to be equal to the potential at the center of the gap to the right of the bars plus a constant to allow for the radial flux ϕ_r. This symmetry plus a constant boundary condition was not available in the finite element package used to model the experimental tests, so an in-house finite element program was written to allow for this boundary condition. The theory used in the in-house program is described in Appendix A, and a listing of the program appears in Appendix B.

Figure 17 is a scale drawing of the finite element mesh used to model the flux-load current interaction. It consists of two bars with a solid iron shield above and solid iron core below. The dimensions of the bar model are shown in Fig. 18. Each triangular element of the model and the node corners of all elements are shown. The currents in the two bars are going in opposite directions. The symmetry plus a constant boundary condition is applied at the center of the space between the bars. The output of the finite element solution is the magnetic potential at each node, from which B and H data can be calculated for the given radial flux and current condition.

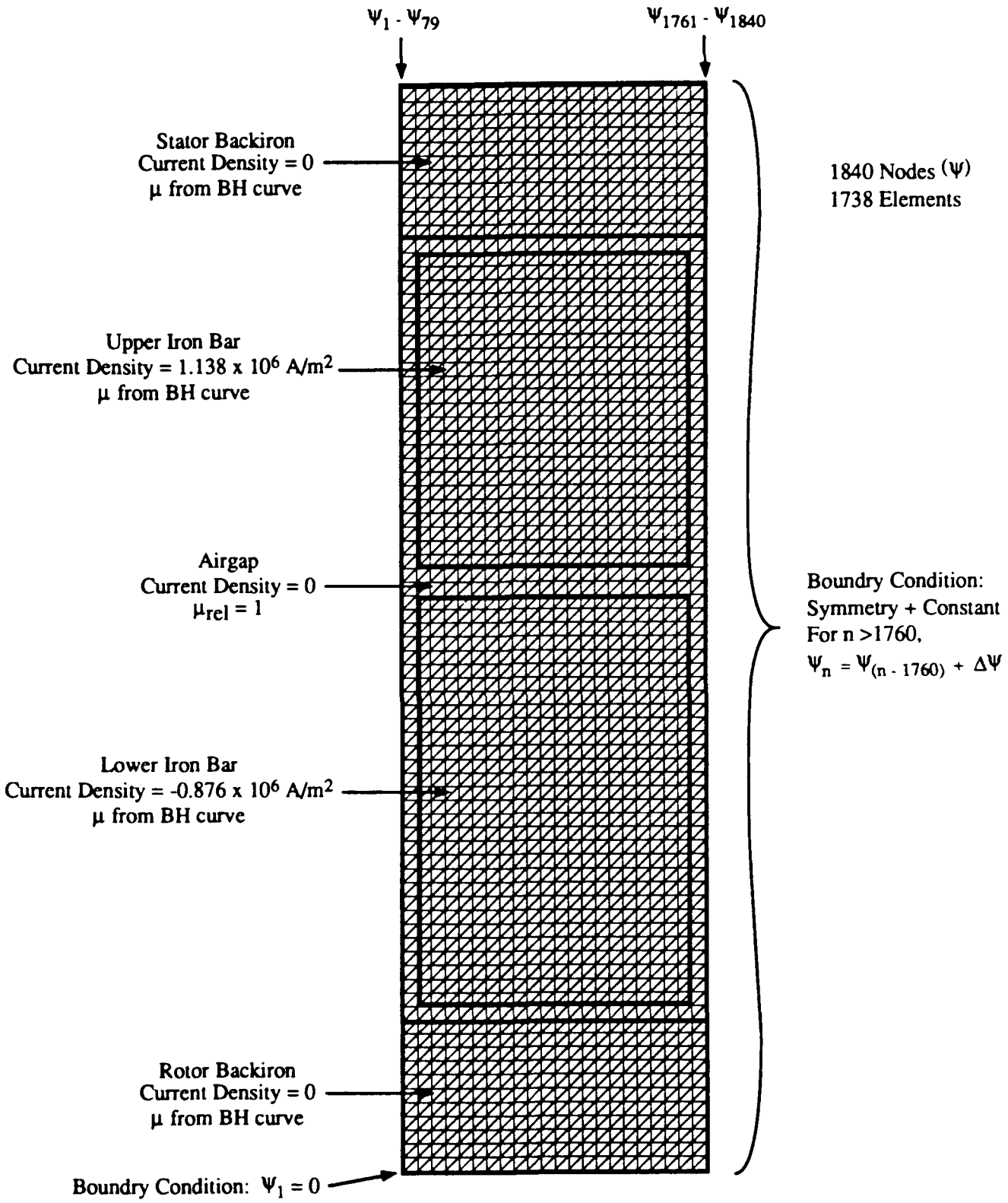


Fig. 17. Finite Element Model Showing the Elements Used and Boundary Conditions for the Case Where the Armature Current is 35 kA. The Corner of Each Triangular Element Represents a Node, Numbered 1 to 1840 Starting From the Lower left Hand Corner.

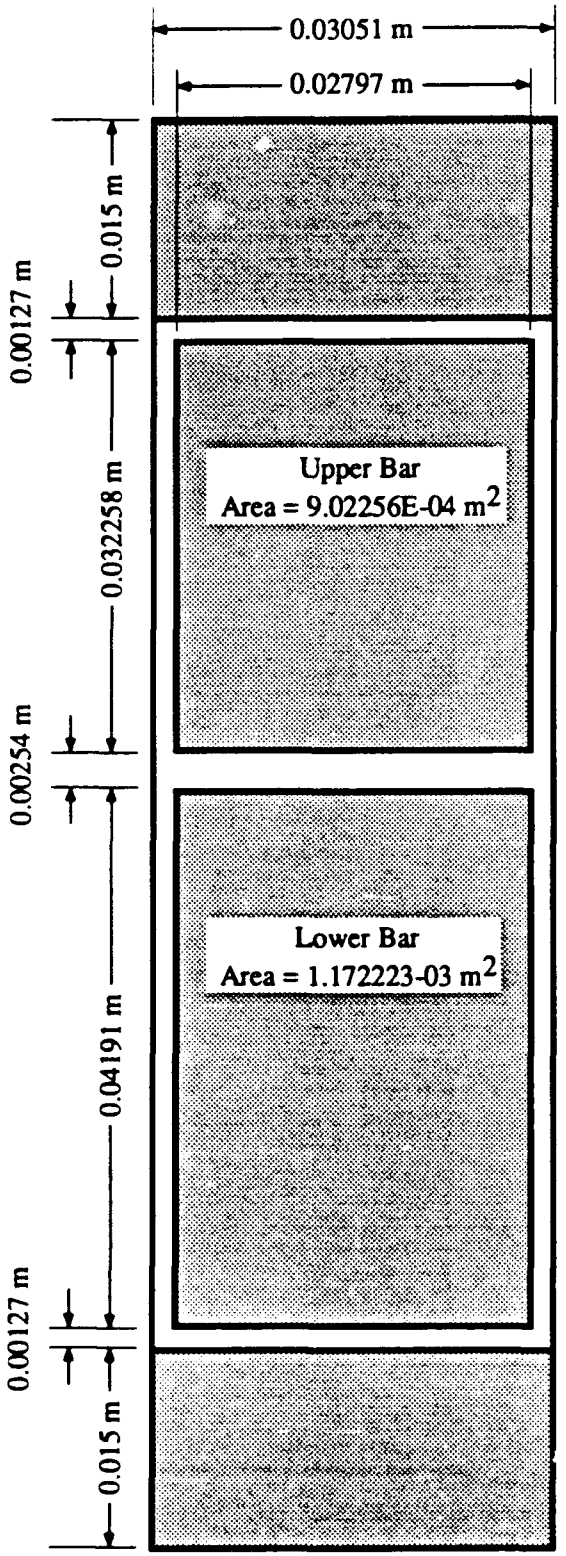


Fig. 18. Dimensions of Finite Element Bar Model.

CALCULATIONS

One parameter that can affect the accuracy of the solution the mesh density. The smaller each individual element, the more accurate the solution, until numerical roundoff becomes an issue. For the computer for which the program was written, a model of 1840 nodes and 1738 elements (700 nodes per meter) was as large as could be comfortably run without memory problems. Figure 19 shows some $\int H \cdot dl$ integration results for various node densities, for an iron bar run with B-radial = .76 Tesla and I-bar = 35000 Amp. By extrapolating Fig. 19, it is estimated that a node density of 700 nodes/meter will produce about a 7% mesh error, which is reasonable considering the errors inherent in measuring consistent BH properties.

Many iron bar finite element models were run using radial flux densities from 0.1 to 8 Tesla, and armature currents from 0 to 100,000 Amps. Some sample flux plots are shown in Fig. 20 for B-radial = .76 Tesla and various current densities in the bars. The flux plots in Fig. 20 show lines of constant potential, the density of lines being equivalent to flux density. Note the familiar fingerprint pattern similar to Fig. 2. Also note that the flux densities in the bars increases significantly even though the flux densities in the core and shield stay constant. The next step was to calculate the equivalent BH characteristics for the iron bars which involves determining the actual H that exists in the iron for any given B-radial seen by the homopolar motor for a particular axial current case. The equivalent BH characteristics were needed for an already existing finite element model of the overall homopolar machine. The overall model included the airgaps above and below the bars, but not the radial spacers (which would require 3 dimensions). Therefore the airgaps are not modeled correctly because the iron bars concentrate the flux underneath and above them, so that $\int H \cdot dl$ through the airgaps is larger than it would be if the flux were evenly distributed over the airgap. This was accounted for by modifing the values of H_{fe} obtained from the flux bar model.

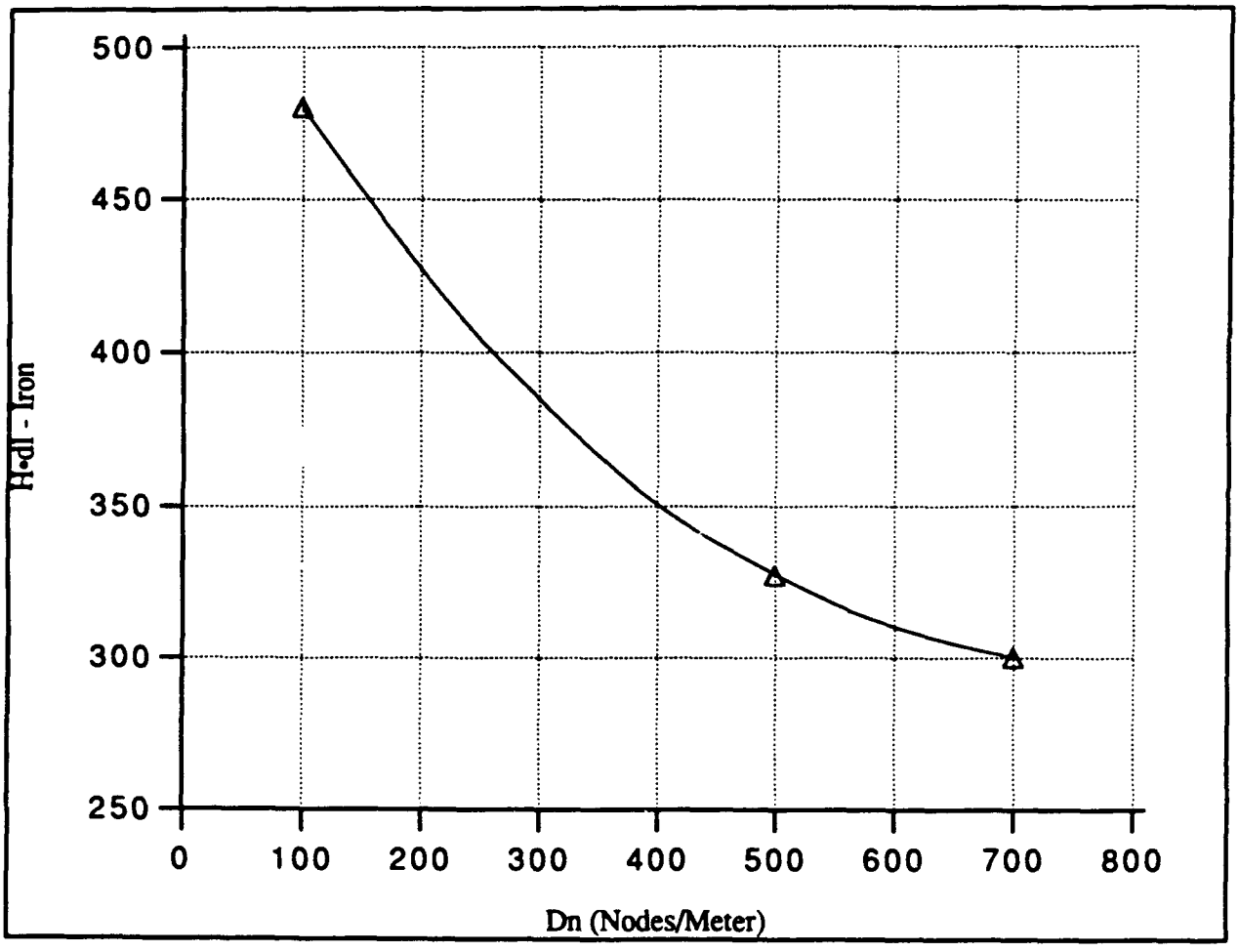
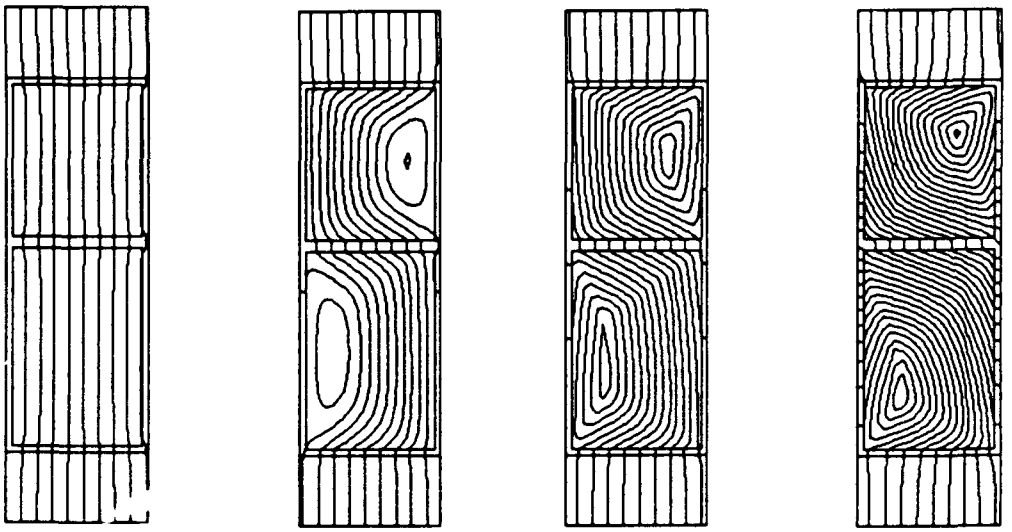


Fig. 19. Effect of Mesh Density of Finite Element Solution for Multibar Case.



Bar Current Density	0 $\frac{A}{in^2}$	40 $\frac{A}{in^2}$	300 $\frac{A}{in^2}$	1850 $\frac{A}{in^2}$
B. Radial	.76T	.76T	.76T	.76T

Fig. 20. Flux Plots in Ferrous Conductors with External Magnetic Fields.

Contours Represent Lines of Constant Potential.
Contour Density is Equivalent to Flux Density.

Average values for magnetic field intensity H can be obtained from the flux bar model by integrating H•dl along a line sement and then dividing by the length of the segment. Using this method the equivalent magnetic intensity H_{fe} for the iron bars was calculated from

$$H_{fe} = \frac{\int_{total} H \cdot dl \; - \; \iint_{airgaps} \left(\frac{B_{radial}}{\mu_0} \right) \cdot dl}{L_{fe}} \tag{13}$$

where

$\int_{total} H \cdot dl$ is integrated from the top of the rotor core to the bottom of the stator shield

$\iint_{airgaps} \left(\frac{B_{radial}}{\mu_0} \right) \cdot dl$ is integrated through the bottom, middle, and upper airgaps

and L_{fe} is the total height of the iron bar segments.

Note that $\int_{total} H \cdot dl$ is the total Amp-turns needed to cross the iron bars and airgaps under actual conditions, and $\iint_{airgaps} \left(\frac{B_{radial}}{\mu_0} \right) \cdot dl = \frac{B_{radial}}{\mu_0} L_{airgaps}$ is the Amp-turns needed for an evenly distributed radial flux to cross the airgaps only which is already included in the overall model. A plot of H_{fe} verses B-radial for all cases is shown in Fig 19 and Table 1. This is the data that was used the homopolar motor overall model.

RESULTS

The plots in Fig. 21 and the data in Table 1 represent the final product for this segment of the project. These curves show that the effective permeability of the iron in the bars is significantly affected by the presence of axial current. These curves were used to represent the bars in a finite element model of the normal conducting homopolar demo motor. The motor analysis will be described in a separate report. However, in the case of the demo motor, the effect of the flux concentration due to axial currents turned out to be less than expected. In the nominal full power case of I_{bar} = 35000 A and B_{radial} = .76 Tesla, the modified BH curves only reduced the overall working flux and torque of the machine by 4%. This is probably because the iron bars are only a short segment of the path of the working flux, and much of the rest of that path includes iron near saturation.

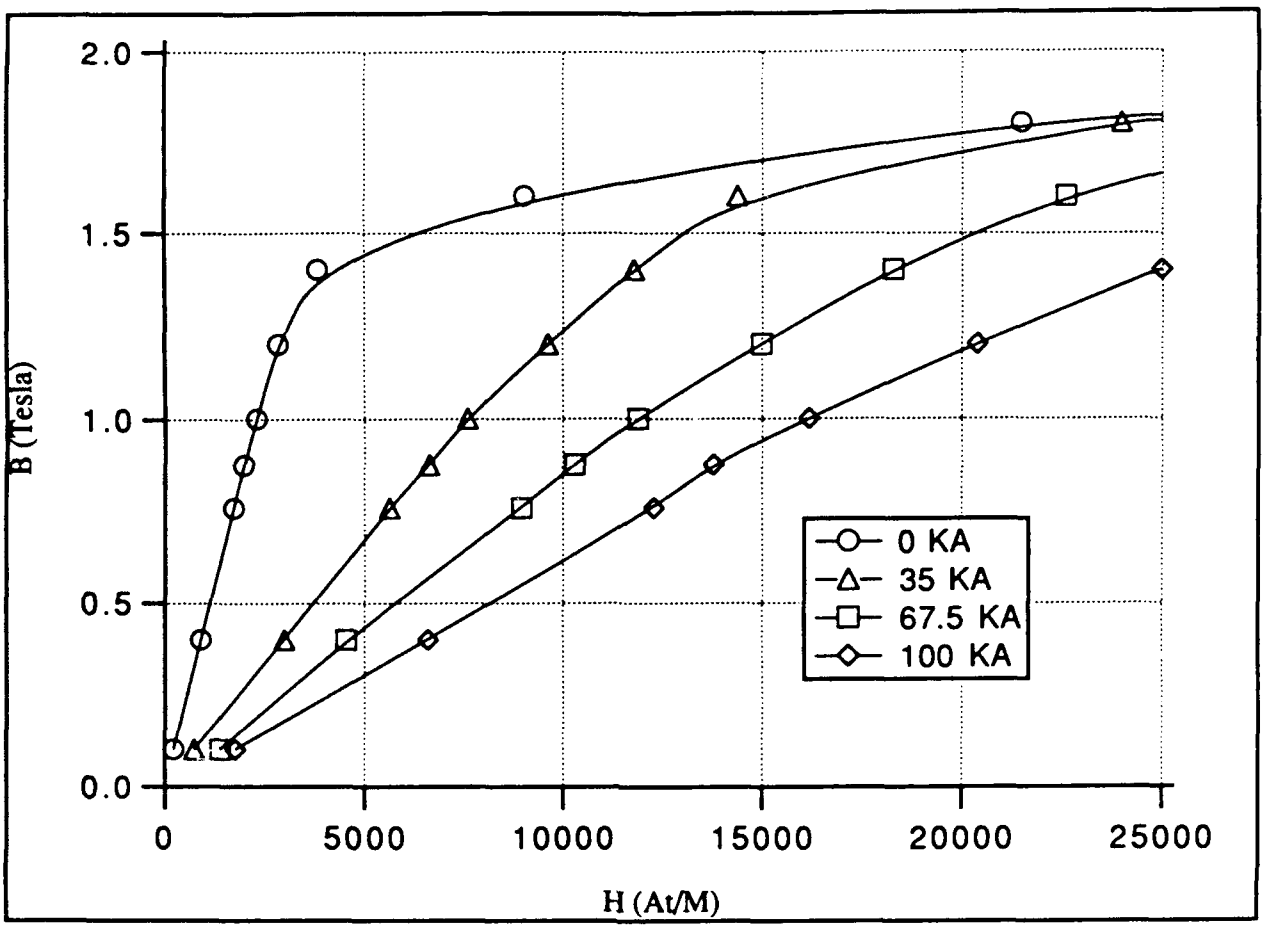


Fig. 21

Effective BH Characteristics for Bar Rotor with Indicated Armature Current

Note to page 21 - Effective BH characteristics for iron bars with axial current include flux concentration due to radial spacers between the bars (H_{fe}). The iron bars and radial spacers are treated as a single material, therefore this data is valid only for the geometry modeled. These curves were used to represent the iron bars in a finite element model of the entire homopolar machine.

Table 1. Data for BH Curves in Fig. 21

Bar Current (A)	B Radial (T)	H Iron+Spacers (At/m)
0K	0.100	227.0
0K	0.400	908.0
0K	0.760	1730.0
0K	0.875	1990.0
0K	1.000	2300.0
0K	1.200	2850.0
0K	1.400	3830.0
0K	1.600	9020.0
0K	1.800	21500.0
0K	2.000	55900.0
0K	8.000	4730000.0
35K	0.100	739.0
35K	0.400	3000.0
35K	0.760	5640.0
35K	0.875	6640.0
35K	1.000	7610.0
35K	1.200	9640.0
35K	1.400	11800.0
35K	1.600	14400.0
35K	1.800	24000.0
35K	2.000	60900.0
35K	8.000	4730000.0
67.5K	0.100	1390.0
67.5K	0.400	4550.0
67.5K	0.760	8970.0
67.5K	0.875	10300.0
67.5K	1.000	11900.0
67.5K	1.200	15000.0
67.5K	1.400	18300.0
67.5K	1.600	22600.0
67.5K	1.800	30700.0
67.5K	2.000	72700.0
67.5K	8.000	4730000.0
100K	0.100	1780.0
100K	0.400	6600.0
100K	0.760	12300.0
100K	0.875	13800.0
100K	1.000	16200.0
100K	1.200	20400.0
100K	1.400	25000.0
100K	1.600	30900.0
100K	1.800	41700.0
100K	2.000	88200.0
100K	8.000	4730000.0

CONCLUSIONS

An accurate, experimentally verified model of the flux-load current interaction has been devised. It shows that the permeability of iron bars changes significantly in the presence of axial currents, and that this effect must be taken into account for an accurate calculation of machine flux and torque. The model is flexible, and can be used for other bar geometries. Although developed for a normal conducting homopolar motor, the in-house finite element program can be useful for other configurations that require the special "symmetry plus a constant" boundary condition, such as modeling a single slot of an AC machine.

APPENDICES

APPENDIX A

FINITE ELEMENT THEORY

Finding the magnetic field produced by electric currents is straightforward as long as the current and magnetic field exist entirely in a uniform linear medium such as air. But once material boundaries and non-linear materials are introduced, magnetic field problems often become too complex to be solved by direct analytical techniques. Such cases often use the finite element method, in which a complex combination of materials and geometry is modeled by many small elements, each consisting of a uniform material of constant permeability. The large problem is then solved by finding the boundary conditions that will solve each of the small element problems simultaneously. Any non-linear effects where the permeability of a material is dependant on the solution are then resolved by iteration.

The first step is to find an equation relating the magnetic potential boundary conditions around a small element of uniform material to the current passing through the element. In two dimensions, the magnetic potential ψ can be found from Poisson's general field equation

$$\frac{1}{\mu}\frac{\partial^2\psi}{\partial^2 x} + \frac{1}{\mu}\frac{\partial^2\psi}{\partial^2 y} + Q = 0 \qquad \text{(A1.1)}$$

were Q is current density, μ is magnetic permeability, and x and y are Cartesian coordinates. Unlike most potential problems, Q and ψ are actually vector quantities. However, in two-dimensional analysis, the direction of Q and ψ are always in the z direction (perpendicular to x and y) so that numerically they can be treated a scalar quantities.

To apply this equation, assume we have a triangular element of a uniform linear material with three nodes i, j, and k. Further, we assume that ψ varies *linearly* over the element, which is a reasonable assumption as long as the element is small compared to the entire problem. The node potential equations for a linear triangular element that satisfy Poisson's equation are found in Segerlind[1]. The form of the potential function is

$$[K]\{\psi\} - \{F\} = \{0\} \qquad \text{(A1.2)}$$

where K is the "stiffness" matrix, ψ is the scalar potential at each node, and F is the "forcing function." The forcing function is calculated as

$$\{F\} = \frac{QA}{3}\begin{Bmatrix}1\\1\\1\end{Bmatrix}$$

---

[1] Larry J. Segerlind, *Applied Finite Element Analysis*,(2nd Edition) Wiley: New York, 1984 pp 56-58, 91-93, 115-120 and 165-171. TA 347 F5 S43 1984

The stiffness matrix is calculated as

$$[K] = \frac{1}{4\mu A}\begin{bmatrix} b_i^2 & b_i b_j & b_i b_k \\ b_i b_j & b_j^2 & b_j b_k \\ b_i b_k & b_j b_k & b_k^2 \end{bmatrix} + \frac{1}{4\mu A}\begin{bmatrix} c_i^2 & c_i c_j & c_i c_k \\ c_i c_j & c_j^2 & c_j c_k \\ c_i c_k & c_j c_k & c_k^2 \end{bmatrix}$$

where A is the area of the triangular element and constants b and c are calculated from the x and y coordinates for nodes i, j, and k as follows:

$$\begin{aligned} b_i &= y_j - y_k & \qquad c_i &= x_k - x_j \\ b_j &= y_k - y_i & \qquad c_j &= x_i - x_k \\ b_k &= y_i - y_j & \qquad c_k &= x_j - x_i \end{aligned}$$

Note that the area of a triangle is

$$A = \frac{1}{2}\left(x_i y_j + x_j y_k + x_k y_i - x_i y_k - x_j y_i - x_k y_j\right)$$

The area is positive if the order of the nodes is chosen so that the surface vector is in the positive z direction. That is, if $\overline{side\ 1} \times \overline{side\ 2}$ and $\overline{side\ 2} \times \overline{side\ 3}$ point in the z direction.

Assuming the permeability μ is constant for any given iteration, Eq A1.2 is a set of three linear algebraic expressions per element relating the potential at the three nodes to the current in the element and the element shape. When elements are side by side there will be more than one equation per node. But since these are linear equations, the various equations for each node can be added together so that there will be as many equations as nodes. The potential of some nodes may be set to a constant or specified by a given function of other nodes to allow for special boundary conditions. For example, in this project, the potential of the nodes at the far right side of the model were set equal to a constant plus the potential of the corresponding nodes at the far left side of the model. The potentials that are not set to define boundary conditions are found by

$$\{\psi\} = [K]^{-1}\{F\}$$

were $[K]^{-1}$ is found numerically. Since $[K]$ is usually a large matrix, finding its inverse involves much computation, and many sophisticated techniques have been derived for efficient solution. However, for the purposes of this project, a basic iteration technique was used since computation speed was not critical and it is the easiest to program.
Once the potential at each node is calculated, the flux density can be calculated from the derivative of the potential

$$B_x = \frac{\partial \psi}{\partial y} \quad \text{and} \quad B_y = \frac{\partial \psi}{\partial x}$$

Numerically this can done by writing an equation for the linear change of ψ over the element where the end potentials are known. That is

$$\psi = \alpha_1 + \alpha_2 x + \alpha_3 y$$

where by definition

$$B_x = \frac{\partial \psi}{\partial y} = \alpha_3$$

$$B_y = \frac{\partial \psi}{\partial x} = \alpha_2$$

The boundary conditions are

$$\psi_i = \alpha_1 + \alpha_2 x_i + \alpha_3 y_i$$
$$\psi_j = \alpha_1 + \alpha_2 x_j + \alpha_3 y_j$$
$$\psi_k = \alpha_1 + \alpha_2 x_k + \alpha_3 y_k$$

Solving for α_1, α_2, and α_3 produces

$$\alpha_2 = B_y = \frac{\psi_i (y_j + y_k) + \psi_j (y_k + y_i) + \psi_k (y_i + y_j)}{2A}$$

$$\alpha_3 = B_x = \frac{\psi_i (x_k + x_j) + \psi_j (x_i + x_k) + \psi_k (x_j + x_i)}{2A}$$

B_x and B_y are then used to find μ for each non-linear element using a lookup table with the BH characteristics for that material. The whole process is then repeated until μ and ψ no longer change.

APPENDIX B

APPENDIX B

Listing of Finite Element Program

This appendix is a listing of the finite element program developed in-house at the Annapolis Detachment, Carderock Division, Naval Surface Warfare Center to calculate flux-current interactions with symmetry plus a constant boundary conditions. The program was written in Microsoft QuickBasic to run on a Macintosh IIcx computer running system 6.0x. (The program needs some minor revision to work with the new system 7). The program is in five parts: 1) user interface and main program, 2) model creator, 3) find node and element connections, 4) write equations, and 5) solve equations. Each part is listed separately with a description of the significance of each part.

THE USER INTERFACE PROGRAM

This is the main program from which the other programs are called. It includes a menu driven interface which calls other programs that create and solve the model. Once created, the model can be examined both graphically and directly. Some examine routines require information that is not available until the equation writer and solver have been called. A special routine added to this program is "Integrate H•dl", which calculates $\int$ H•dl between specified points. This routine was used to calculate the effective BH characteristics of the iron with axial currents.

```
' Finite Element Main Program 9/15/89

COMMON WhoAmI$,FileName$,Printf%,menu0%,menu1%
WhoAmI$="FE 8/21 shell.s"
Solve$="Solve Mesh.s"
Connect$="Find Node Connections.s"
WriteEqu$="Write Equations.s"
IF SYSTEM(4) THEN ' SYSTEM(4)=True if program is compiled
  WhoAmI$=""
  Solve$="Solve Mesh"
  Connect$="Find Node Connections"
  WriteEqu$="Write Equations"
  FOR adr%=&H911 TO &H910 + PEEK(&H910) ' Looks up name of Application
    WhoAmI$=WhoAmI$+CHR$(PEEK(adr%))
  NEXT adr%
END IF
PRINT "**"+WhoAmI$+"**"

OPTION BASE 0
DIM Nu%(2000),Vc(2000),x(2000),y(2000),v(2000)
DIM Ni%(2000),Nj%(2000),Nk%(2000),Nl%(2000)
DIM mate%(2000),Type%(2000),Curd(2000),u1(2000),u2(2000)
DIM Na%(2000),Nb%(2000),Nc%(2000),Nd%(2000)
DIM plot(200)

MENU 1,0,1,"File"
MENU 1,1,1,"Open"
```

```
MENU 1,2,0,"Save As"
MENU 1,3,1,"Quit" : CmdKey 1,3,"Q"
MENU 2,0,1,"Create"
MENU 2,1,1,"Run External Program Create Mesh"
MENU 2,2,0,"-"
MENU 2,3,1,"Find External Program"
MENU 2,4,1,"Re-Write Equations"
MENU 3,0,1,"Solve"
MENU 3,1,1,"Solve Mesh"
MENU 4,0,1,"Examine"
MENU 4,1,1,"Examine Entire File"
MENU 4,2,1,"Examine Node Positions"
MENU 4,3,1,"Examine Node Potentials"
MENU 4,4,1,"Examine Node Connections"
MENU 4,5,1,"Examine Element Compostion"
MENU 4,6,1,"Examine Element Properties"
MENU 4,7,1,"Examine Element Connections"
MENU 4,8,1,"Examine Solution Equations"
MENU 4,9,0,"-"
MENU 4,10,1,"Print to Screen"
MENU 4,11,1,"Print to File"
MENU 4,12,1,"Cancel Printing"
MENU 5,0,1,"Plots"
MENU 5,1,1,"Material Outline"
MENU 5,2,1,"Mesh"
MENU 5,3,1,"Regions"
MENU 5,4,1,"Flux"
MENU 6,0,1,"Integrate"
MENU 6,1,1,"H•dl" : CmdKey 6,1,"H"
'MENU 6,2,1,"print V70 equations"

u0=.000001256637#
MeshFile%=0  ' Nothing has been read from the file yet (even if returning)

CLOSE #2          ' Always return with screen as output for examine menu
OPEN "SCRN:" FOR OUTPUT AS #2  ' May want to eliminate later

ReturnFn%=1
'PRINT  "***";FileName$;"***";menu0%;menu1%
IF FileName$<>"" THEN
   MultiCommand%=1 : menu0p%=menu0% : menu1p%=menu1%
   IF menu0%=2 AND menu1%<4 THEN MultiCommand%=0
   IF menu0%=0 THEN MultiCommand%=0  ' Set by Solve Mesh prog
   ReturnFn%=0
   menu0%=1 : menu1%=1 : GOTO while2
END IF

loop: menu0p%=0
menu1p%=0
menu0%=MENU(0)
mselect: menu1%=MENU(1)
MultiCommand%=0
level%=0
while2: WHILE menu0% OR MultiCommand%
   scase: SELECT CASE menu0%

      '    ************ CASE 1  Open File *************

    CASE 1  ' File Menu selected
     SELECT CASE menu1%

       CASE 1
        ' Selected Open.
        IF ReturnFn% THEN FileName$=FILES$(1,"MESH")
```

```
    IF FileName$="" THEN
      MultiCommand%=0
    ELSE
      changeCursor 4
      ReturnFn%=1
      CLOSE #1          ' Close File if already open
      OPEN "R",#1,FileName$,8
      WINDOW 1, FileName$
      FIELD #1, 2 AS aa2$, 2 AS ab2$, 2 AS ac2$, 2 AS ad2$
      FIELD #1, 2 AS ba2$, 2 AS bb2$, 4 AS bc4$
      FIELD #1, 4 AS ca4$, 4 AS cb4$
      GET #1, 1
      Nodes%=CVI(aa2$) : Elements%=CVI(ab2$)
      Nodes1%=Nodes%+1 : Elements1%=Elements%+1
      TestV%=CVI(ac2$) : TestE%=CVI(ad2$)
      PRINT "Nodes =";Nodes%;"     Elements =";Elements%
      PRINT TestV%,TestE%
      nuvcF%=0 : inuvc%=1
      xyF%=0 : ixy%=1 : xypt&=1+Nodes%
      vF%=0 : iv%=1 : vpt&=1+2*Nodes%
      nijklF%=0 : inijkl%=1 : nijklpt&=1+3*Nodes%
      mcurdF%=0 : imcurd%=1 : mcurdpt&=1+3*Nodes%+Elements%
      typeF%=0 : itype%=1 : typept&=1+3*Nodes%+Elements%
      uF%=0 : iu%=1 : upt&=1+3*Nodes%+2*Elements%
      nregF%=0 : ireg%=1 : regpt&=1+3*Nodes%+3*Elements% : nreg%=0
      nabcdF%=0 : inabcd%=1 : nabcdpt&=1+3*Nodes%+4*Elements%
      equpt&=2+3*Nodes%+5*Elements%
      MeshFile%=1
      MENU 1,2,1,"Save As"
    END IF

  CASE 2
    ' Selected Save As.
    NewMFile$=FILES$(0,"New MESH Filename")
    IF NewMFile$<>"" THEN
      changeCursor 4
      copyFile FileName$, NewMFile$
      NAME NewMFile$ AS NewMFile$, "MESH"
      FileName$=NewMFile$
      WINDOW 1, FileName$
    END IF

  CASE 3
    ' Selected Quit.  Be sure to save current results
    STOP

  CASE ELSE
    PRINT "Error:  Not Expecting Menu 1 Item";MENU(1);"to be selected." : INPUT x : STOP
  END SELECT

'    ************* CASE 2  Run Model Creator **************

CASE 2  ' Create Menu selected
  SELECT CASE menu1%

  CASE 1
    IF SYSTEM(4) THEN mne$=FILES$(1,"Pext") ELSE mne$=FILES$(1,"TEXT")
    IF mne$<>"" THEN CHAIN mne$

  CASE 3
    ' Change File Type to Pext (external Program)
    IF SYSTEM(4) THEN mne$=FILES$(1,"APPL") ELSE mne$=FILES$(1,"TEXT")
    IF mne$<>"" THEN
      IF SYSTEM(4) THEN NAME mne$ AS mne$, "Pext"
```

```
      CHAIN mne$
    END IF

  CASE 4
    ' Manual re-write equations
    IF MeshFile% THEN
      GOSUB findPnode
      menu0%=2 : menu1%=5 : GOTO scase
    ELSE
      PRINT "No File Open."
    END IF

  CASE 5
    menu0%=2 : menu1%=6
    CHAIN Connect$

  CASE 6
    menu0%=2 : menu1%=7
    CHAIN WriteEqu$

  CASE 7
    PRINT "Completed"
    MultiCommand%=0
    GOTO Idle

  CASE ELSE
    PRINT "Error: Not Expecting Menu 2 Item";MENU(1);"to be selected."
    INPUT x : STOP
  END SELECT

  '    ************ CASE 3 Run Equation Solver *************

CASE 3 ' Solve Menu selected
  ' IF TestV%>3 THEN PRINT "Convergence already completed" : GOTO Idle
  changeCursor 4
  IF MeshFile%=0 THEN
    MultiCommand%=1
    menu0p%=3
    menu1p%=1
    menu0%=1 : menu1%=1 : GOTO scase
  END IF
  MultiCommand%=0
  IF (TestV% AND 2)=0 THEN GOSUB findPnode
  IF (TestV% AND 1)=0 THEN GOSUB readv ' Set all potentials to Zero. (file may have junk!)
  ' Do this first because vpt& and other similar variables lost after CHAIN to Connect$
  menu0%=3 : menu1%=1
  IF (TestE% AND 1)=0 THEN CHAIN Connect$
  'IF TestE% =0 OR TestE%=2 THEN CHAIN Connect$
  IF (TestE% AND 2)=0 THEN CHAIN WriteEqu$
  'IF TestE%<2 THEN CHAIN WriteEqu$
  menu0%=0 : menu1%=0
  CHAIN Solve$

  '    ************ CASE 4 Examine File Contents *************

CASE 4 ' Examine Menu selected

  IF menu1%<9 AND MultiCommand%=0 THEN
    IF Printf% THEN changeCursor 4
    ' IF MeshFile%=0 THEN PRINT "File Not Open." : GOTO Idle
    IF MeshFile%=0 THEN
      MultiCommand%=1
      menu0p%=4
      menu1p%=menu1%
```

```
        menu0%=1 : menu1%=1 : GOTO scase
      END IF
      IF Nodes%=0 AND Elements%=0 THEN PRINT "File Is Empty." : GOTO Idle
    END IF
    SELECT CASE menu1%

     CASE 1 ' Examine Whole File
      MultiCommand%=1
      menu0p%=4
      menu1p%=1
      IF level%=0 THEN level%=1 : menu1%=4 : GOTO scase
      IF level%=1 THEN level%=2 : menu1%=2 : GOTO scase
      IF level%=2 THEN level%=3 : menu1%=3 : GOTO scase
      IF level%=3 THEN level%=4 : menu1%=5 : GOTO scase
      IF level%=4 THEN level%=5 : menu1%=6 : GOTO scase
      IF level%=5 THEN level%=6 : menu1%=7 : GOTO scase
      IF level%=6 THEN level%=7 : menu1%=8 : GOTO scase
      MultiCommand%=0

     CASE 2 ' Examine Node Positions
      IF xyF%=0 THEN GOSUB readxy
      PRINT #2, "Xmin =";xmin;TAB(20);"Xmax =";xmax
      PRINT #2, "Ymin =";ymin;TAB(20);"Ymax =";ymax
      PRINT #2, "Node";TAB(10);"X";TAB(26);"Y"
      FOR i%=1 TO Nodes%
        PRINT #2, i%;TAB(9);x(i%);TAB(25);y(i%)
        menu0%=MENU(0) : IF menu0%>0 THEN mselect
      NEXT i%

     CASE 3 ' Examine Node Potentials
      IF vF%=0 THEN GOSUB readv
      PRINT #2, "Vmin =";vmin;TAB(22);"Vmax =";vmax
      PRINT #2, "Node";TAB(10);"V"
      FOR i%=1 TO Nodes%
        PRINT #2, i%;TAB(9);v(i%)
        menu0%=MENU(0) : IF menu0%>0 THEN mselect
      NEXT i%

     CASE 4 ' Examine Node Connections
      IF nuvcF%=0 THEN GOSUB readnuvc
      PRINT #2, "Node";TAB(10);"Nu";TAB(19);"Vc"
      FOR i%=1 TO Nodes%
        PRINT #2, i%;TAB(9);Nu%(i%);TAB(18);Vc(i%)
        menu0%=MENU(0) : IF menu0%>0 THEN mselect
      NEXT i%

     CASE 5 ' Examine Element Compostion
      IF nijklF%=0 THEN GOSUB readnijkl
      ' IF nregF%=0 THEN GOSUB Findnreg
      PRINT #2,
"Element";TAB(10);"Region";TAB(20);"Ni";TAB(29);"Nj";TAB(38);"Nk";TAB(47);"Nl"
      GET #1, regpt&
      FOR i%=1 TO Elements%
        GET #1
        Region%=CVI(aa2$)
        PRINT #2,
i%;TAB(9);Region%;TAB(19);Ni%(i%);TAB(28);Nj%(i%);TAB(37);Nk%(i%);TAB(46);Nl%(i%)
        menu0%=MENU(0) : IF menu0%>0 THEN mselect
      NEXT i%

     CASE 6 ' Examine Element Properties
      IF mcurdF%=0 THEN GOSUB readmcurd
      IF uF%=0 THEN GOSUB readu
      IF typeF%=0 THEN GOSUB FindType
```

```
      PRINT #2,
"Element";TAB(10);"Mate";TAB(16);"Type";TAB(22);"Curd";TAB(37);"Perm1";TAB(52);"Perm2"
      FOR i%=1 TO Elements%
        PRINT #2, i%;TAB(9);mate%(i%);TAB(15);Type%(i%);TAB(21);Curd(i%);
        PRINT #2, TAB(36);u1(i%);TAB(51);u2(i%)
        menu0%=MENU(0) : IF menu0%>0 THEN mselect
      NEXT i%

    CASE 7 ' Examine Element Connections
      IF nabcdF%=0 THEN GOSUB FindNabcd
      PRINT #2, "Element";TAB(10);"xNa";TAB(19);"xNb";TAB(28);"xNc";TAB(37);"xNd"
      FOR i%=1 TO Elements%
        PRINT #2, i%;TAB(9);Na%(i%);TAB(18);Nb%(i%);TAB(27);Nc%(i%);TAB(36);Nd%(i%)
        menu0%=MENU(0) : IF menu0%>0 THEN mselect
      NEXT i%

    CASE 8 ' Examine Solution Equations
      IF TestE%<2 THEN CHAIN WriteEqu$
      GET #1, equpt& ' StartVequ& StartBequ&
      StartVequ&=CVL(ca4$)
      StartBequ&=CVL(cb4$)
      ' PRINT "From equpt& =";equpt&;" Find StartVequ& =";StartVequ&
      GET #1, StartVequ& ' inodes%, blank%, StartIequ&
      inodes%=CVI(ba2$)
      StartDequ&=CVL(bc4$)
      PRINT #2, inodes%;"Set Node Potentials starting at";StartVequ&
      FOR i%=1 TO inodes%
        GET #1
        Vi%=CVI(ba2$) : c1=CVS(bc4$)
        PRINT #2, "V"+MID$(STR$(Vi%),2)+"=";c1
        menu0%=MENU(0) : IF menu0%>0 THEN mselect
      NEXT i%

      GET #1, StartDequ& ' inodes%, blank%, StartDequ&
      inodes%=CVI(ba2$)
      StartIequ&=CVL(bc4$)
      PRINT #2, inodes%;"Dependent Node Equations starting at";StartDequ&
      FOR i%=1 TO inodes%
        GET #1
        Vi%=CVI(ba2$) : Vn%=CVI(bb2$) : c1=CVS(bc4$)
        PRINT #2, "V"+MID$(STR$(Vi%),2)+"=V"+MID$(STR$(Vn%),2);
        IF c1<0 THEN PRINT #2, c1 ELSE PRINT #2, "+"+MID$(STR$(c1),2)
        menu0%=MENU(0) : IF menu0%>0 THEN mselect
      NEXT i%

      GET #1, StartIequ& ' inodes%, blank%, StartDequ&
      inodes%=CVI(ba2$)
      StartBequ&=CVL(bc4$)
      PRINT #2, inodes%;"Independent Node Equations starting at";StartIequ&
      FOR i%=1 TO inodes%
        GET #1
        Vi%=CVI(ba2$) : Nline%=CVI(bb2$) : c1=CVS(bc4$)
        IF c1=0 THEN C$="" ELSE C$=STR$(c1)
        PRINT #2, "V"+MID$(STR$(Vi%),2)+"=("+C$;
        d$=""
        FOR j%=1 TO Nline%
          GET #1
          Vn%=CVI(ba2$) : un%=CVI(bb2$) : kij=CVS(bc4$)
          IF kij=1 THEN k$="" ELSE k$=STR$(kij) : IF kij>=0 THEN MID$(k$,1,1)="+"
          IF un%=0 THEN u$="" ELSE u$="u"+MID$(STR$(un%),2)
          IF k$="" THEN u$="+"+u$ ELSE u$="*"+u$
          v$="V"+MID$(STR$(Vn%),2)
          IF u$<>"+" AND u$<>"*" THEN v$="*"+v$
          PRINT #2, k$+u$+v$;
```

```
            IF un%=0 AND kij=1 THEN d$=d$+"1" ELSE d$=d$+k$+u$
          NEXT j%
          IF d$="1" THEN PRINT #2, ")" ELSE PRINT #2, ")/("+d$+")"
          menu0%=MENU(0) : IF menu0%>0 THEN mselect
        NEXT i%

        GET #1, StartBequ& ' inodes%, blank%, Endequ&
        inodes%=CVI(ba2$)
        PRINT #2, inodes%;"Flux Density Equations  starting at";StartBequ&
        FOR i%=1 TO inodes%
          GET #1
          nelem%=CVI(aa2$) : v1i%=CVI(ab2$) : v1j%=CVI(ac2$) : v1k%=CVI(ad2$)
          GET #1
          ci=CVS(ca4$) : cj=CVS(cb4$) : ck=-ci-cj
          GET #1
          bi=CVS(ca4$) : bj=CVS(cb4$) : bk=-bi-bj
          PRINT #2, "Bx-
E"+MID$(STR$(nelem%),2)+"=";ci;"*V"+MID$(STR$(v1i%),2)+"+";cj;"*V"+MID$(STR$(v1j%),2)+"+
";ck;"*V"+MID$(STR$(v1k%),2)
          PRINT #2, "By-
E"+MID$(STR$(nelem%),2)+"=";bi;"*V"+MID$(STR$(v1i%),2)+"+";bj;"*V"+MID$(STR$(v1j%),2)+"+
";bk;"*V"+MID$(STR$(v1k%),2)
          menu0%=MENU(0) : IF menu0%>0 THEN mselect
        NEXT i%

      CASE 10
        CLOSE #2
        OPEN "SCRN:" FOR OUTPUT AS #2
        Printf%=0

      CASE 11
        mne$=FILES$(0,"Disk File for Screen Dump")
        IF mne$<>"" THEN
          CLOSE #2
          OPEN mne$ FOR OUTPUT AS #2
          Printf%=1
        END IF

      CASE 12
        ' Cancel Print.    (goto Idle)
        MultiCommand%=0

      CASE ELSE
        PRINT "Menu 4 Not Finished Yet" : INPUT x : STOP
    END SELECT

    IF menu1%<9 AND MultiCommand%=0 THEN
      PRINT #2, "End of File."
    END IF

    '    ************ CASE 5  Plot Material Outlines, Mesh, and Flux  (constant potential)
*************

   CASE 5 ' Plot selected
    IF MeshFile%=0 THEN
      MultiCommand%=1
      menu0p%=5
      menu1p%=menu1%
      menu0%=1 : menu1%=1 : GOTO scase
    ELSE
      IF PlotFluxF%=0 THEN MultiCommand%=0 ' Draw Material Outline with fluxplot
    END IF
    IF Nodes%=0 AND Elements%=0 THEN PRINT "File Is Empty." : GOTO Idle
    changeCursor 4
```

```
IF nabcdF%=0 AND (menu1%=1 OR menu1%=4) THEN GOSUB FindNabcd
IF xyF%=0 THEN GOSUB readxy
IF nijklF%=0 THEN GOSUB readnijkl

SELECT CASE menu1%
 CASE 1 ' Draw Material Outline

  IF mcurdF%=0 THEN GOSUB readmcurd
  IF typeF%=0 THEN GOSUB FindType
  frame xmin,xmax,ymin,ymax,-.9
  CLS
  PRINT xmin,xmax,ymin,ymax
  mate%(0)=-1
  FOR i%=1 TO Elements%
    m1%=mate%(i%)
    IF mate%(Na%(i%))<>m1% THEN draw x(Ni%(i%)), y(Ni%(i%)), x(Nj%(i%)), y(Nj%(i%))
    IF mate%(Nb%(i%))<>m1% THEN draw x(Nj%(i%)), y(Nj%(i%)), x(Nk%(i%)), y(Nk%(i%))
    IF mate%(Nc%(i%))<>m1% THEN draw x(Nk%(i%)), y(Nk%(i%)), x(Nl%(i%)), y(Nl%(i%))
    IF mate%(Nd%(i%))<>m1% AND Type%(i%)>=0 THEN draw x(Nl%(i%)), y(Nl%(i%)),
x(Ni%(i%)), y(Ni%(i%))
    menu0%=MENU(0) : IF menu0%>0 THEN mselect
  NEXT i%

 CASE 2 ' Draw Mesh

  IF typeF%=0 THEN GOSUB FindType
  frame xmin,xmax,ymin,ymax,-.9
  CLS
  PRINT xmin,xmax,ymin,ymax
  FOR i%=1 TO Elements%
    n1%=Ni%(i%) : n2%=Nj%(i%) : n3%=Nk%(i%) : n4%=Nl%(i%)
    SELECT CASE Type%(i%)
     CASE 0
      draw x(n1%), y(n1%), x(n2%), y(n2%)
      drawto x(n3%), y(n3%)
      drawto x(n4%), y(n4%)
      drawto x(n1%), y(n1%)
     CASE 1
      draw x(n4%), y(n4%), x(n1%), y(n1%)
      drawto x(n2%), y(n2%)
      drawto x(n3%), y(n3%)
      drawto x(n4%), y(n4%)
      drawto x(n2%), y(n2%)
     CASE 2
      draw x(n1%), y(n1%), x(n2%), y(n2%)
      drawto x(n3%), y(n3%)
      drawto x(n4%), y(n4%)
      drawto x(n1%), y(n1%)
      drawto x(n3%), y(n3%)
    END SELECT
    menu0%=MENU(0) : IF menu0%>0 THEN mselect
  NEXT i%

 CASE 3 ' Draw Region

  IF nregF%=0 THEN GOSUB Findnreg
  PRINT "Draw Region not yet finished"

 CASE 4 ' Draw Flux Plot

  IF vF%=0 THEN GOSUB readv
  IF vmax-vmin=0 THEN PRINT "No Potentials Defined" : GOTO Idle

  IF PlotFluxF%=0 THEN
```

```
    INPUT "Flux lines to plot",Nline%
    IF Nline%>200 THEN Nline%=200
    IF Nline%=0 THEN Nline%=10
    PlotFluxF%=1
    MultiCommand%=1
    menu0p%=5
    menu1p%=4
    menu0%=5 : menu1%=1 : GOTO scase
  ELSE
    PlotFluxF%=0
    MultiCommand%=0
END IF

FOR i%=1 TO Nline%
  plot(i%)=vmin+(vmax-vmin)*(i%-.5)/Nline%
NEXT i%

FOR i%=1 TO Elements%
  n1%=Ni%(i%) : n2%=Nj%(i%) : n3%=Nk%(i%) : n4%=Nl%(i%)
  v1=v(n1%) : v2=v(n2%) : v3=v(n3%) : v4=v(n4%)
  n5%=n1% : n6%=n2% : n7%=n3% : n8%=n4%
  v5=v1 : v6=v2 : v7=v3 : v8=v4
  IF v1>v6 THEN SWAP v1, v6 : SWAP n1%, n6%
  IF v2>v7 THEN SWAP v2, v7 : SWAP n2%, n7%
  IF v3>v8 THEN SWAP v3, v8 : SWAP n3%, n8%
  IF v4>v5 THEN SWAP v4, v5 : SWAP n4%, n5%
  FOR j%=1 TO Nline%
    test%=0
    v=plot(j%)

    IF v>v1 AND v<v6 THEN
      test%=test%+1
      IF test%=1 THEN
        Inter v1,x(n1%),v6,x(n6%),v,x1
        Inter v1,y(n1%),v6,y(n6%),v,y1
      ELSE
        Inter v1,x(n1%),v6,x(n6%),v,x2
        Inter v1,y(n1%),v6,y(n6%),v,y2
        draw x1,y1,x2,y2
        test%=0
      END IF
    END IF

    IF v>v2 AND v<v7 THEN
      test%=test%+1
      IF test%=1 THEN
        Inter v2,x(n2%),v7,x(n7%),v,x1
        Inter v2,y(n2%),v7,y(n7%),v,y1
      ELSE
        Inter v2,x(n2%),v7,x(n7%),v,x2
        Inter v2,y(n2%),v7,y(n7%),v,y2
        draw x1,y1,x2,y2
        test%=0
      END IF
    END IF

    IF v>v3 AND v<v8 THEN
      test%=test%+1
      IF test%=1 THEN
        Inter v3,x(n3%),v8,x(n8%),v,x1
        Inter v3,y(n3%),v8,y(n8%),v,y1
      ELSE
        Inter v3,x(n3%),v8,x(n8%),v,x2
        Inter v3,y(n3%),v8,y(n8%),v,y2
```

```
            draw x1,y1,x2,y2
            test%=0
          END IF
        END IF

        IF v>v4 AND v<v5 THEN
          test%=test%+1
          IF test%=1 THEN
            Inter v4,x(n4%),v5,x(n5%),v,x1
            Inter v4,y(n4%),v5,y(n5%),v,y1
          ELSE
            Inter v4,x(n4%),v5,x(n5%),v,x2
            Inter v4,y(n4%),v5,y(n5%),v,y2
            draw x1,y1,x2,y2
            test%=0
          END IF
        END IF

      NEXT j%
    NEXT i%

  END SELECT

  '    ********** CASE 6   Integrate H • dl  **********

CASE 6  ' Integrate selected

  IF MeshFile%=0 THEN
    MultiCommand%=1
    menu0p%=6
    menu1p%=1
    menu0%=1 :  menu1%=1 :  GOTO scase
  END IF
  changeCursor 4
  IF vF%=0 THEN GOSUB readv
  IF vmax-vmin=0 THEN PRINT "No Potentials Defined" :  GOTO Idle

  ' Trashing info in Nu%(), Vc(), plot()
  IF xyF%=0 THEN GOSUB readxy
  IF nijklF%=0 THEN GOSUB readnijkl
  IF typeF%=0 THEN GOSUB FindType
  IF uF%=0 THEN GOSUB readu

  Hagain: WINDOW OUTPUT 1
  IF PlotFluxF%=0 THEN
    PlotFluxF%=1
    MultiCommand%=1
    menu0p%=6
    menu1p%=1
    menu0%=5 :  menu1%=1 :  GOTO scase
   ELSE
    PlotFluxF%=0
  END IF
  MultiCommand%=0

  INITCURSOR
  Hdl#=0#
  WINDOW 2,,(350,30)-(500,200),4  ' 512,340  bottom right
  CLS
  PRINT "Integrate H•dl"
  PRINT "(x1,y1) to (x2,y2)"
  INPUT "x1";x2
  INPUT "y1";y2
  mpoint: x1=x2 :  y1=y2
```

```
    INPUT "x2";x2
    INPUT "y2";y2
    WINDOW OUTPUT 1  ' Don't cover window 2

    CALL defnodecond  ' Define position definition in Nu% for each node
    IF menu0%>0 THEN mselect
    FOR n%=1 TO 2
      begin%=1
      WHILE begin%>0
        getdlelem n%,begin%,elem%,xh1,yn1,xh2,yh2,factor
        IF elem%>0 THEN

          draw xh1,yh1,xh2,yh2
          x1i=x(n11%) : y1i=y(n11%)
          x1j=x(n12%) : y1j=y(n12%)
          x1k=x(n13%) : y1k=y(n13%)
          a14=1/((x1i-x1j)*(y1i-y1k)+(y1i-y1j)*(x1k-x1i))  ' 1/(2*Area)  positive direction.
          b1i=(y1j-y1k)*a14 : b1j=(y1k-y1i)*a14 : b1k=-b1i-b1j
          c1i=(x1k-x1j)*a14 : c1j=(x1i-x1k)*a14
          Bx=(v(n11%)-v(n13%))*c1i+(v(n12%)-v(n13%))*c1j
          By=(v(n13%)-v(n11%))*b1i+(v(n13%)-v(n12%))*b1j
          IF n%=1 THEN
            Hx=Bx*u1(elem%)  ' u1 is 1/μr (inverse of relative permubility
            Hy=By*u1(elem%)
           ELSE
            Hx=Bx*u2(elem%)
            Hy=By*u2(elem%)
          END IF
          Hdl#=Hdl#+(Hx*(xh2-xh1)+Hy*(yh2-yh1))*factor
          'PRINT "Bx =";Bx
          'PRINT "By =";By
        END IF
        begin%=elem%+1
      WEND
    NEXT n%
    WINDOW OUTPUT 2
    PRINT "H•dl=";CSNG(Hdl#/u0)
    INPUT "Continue (y ,n or r)";x$
    IF UCASE$(MID$(x$,1,1))="R" THEN Hagain  ' Start over with another integration
    IF UCASE$(MID$(x$,1,1))<>"N" THEN mpoint  ' Continue integration to another point
    MultiCommand%=0
    WINDOW 1

    '    ************ CASE ELSE *************

   CASE ELSE
    PRINT "Opps !!!  Something wierd has happened.  Hit RETURN and proceed with caution."
    MultiCommand%=0
    INPUT x
  END SELECT

  Idle: IF MultiCommand% THEN
    menu0%=menu0p%
    menu1%=menu1p%
   ELSE
    menu0%=0
    INITCURSOR
    MENU 1,1,1,"Open"  ' Deselect Menu Items  --- Idle loop
  END IF

WEND
' Idle Stuff
IF MeshFile% THEN   ' Read File while waiting for user input.
  IF xyF%=0 THEN GOSUB readxy
```

```
   IF nijklF%=0 THEN GOSUB readnijkl
   IF vF%=0 THEN GOSUB readv
   IF mcurdF%=0 THEN GOSUB readmcurd
   IF uF%=0 THEN GOSUB readu
   IF typeF%=0 THEN GOSUB FindType
   IF nabcdF%=0 AND (TestE%=1 OR TestE%=3) THEN GOSUB FindNabcd
   IF nuvcF%=0 THEN GOSUB readnuvc
   IF nregF%=0 THEN GOSUB Findnreg
END IF
GOTO loop

' ################### SUBS ######################

readnuvc: ' Node     Nu       Vc"
' PRINT "Reading Nu, Vc"
WHILE inuvc%<Nodes1%
   GET #1, inuvc%+1
   Nu%(inuvc%)=CVI(ba2$)
   Vc(inuvc%)=CVS(bc4$)
   inuvc%=inuvc%+1
   menu0%=MENU(0) : IF menu0%>0 THEN mselect
WEND
inuvc%=1
nuvcF%=1
RETURN

readxy: ' Node     X     Y"
' PRINT "Reading X, Y"
' xypt&=1+Nodes%
IF ixy%=1 THEN
   GET #1, 1+xypt&
   xmin=CVS(ca4$) : xmax=CVS(ca4$)
   ymin=CVS(cb4$) : ymax=CVS(cb4$)
END IF
WHILE ixy%<Nodes1%
   GET #1, ixy%+xypt&
   x=CVS(ca4$)
   y=CVS(cb4$)
   IF x<xmin THEN xmin=x
   IF y<ymin THEN ymin=y
   IF x>xmax THEN xmax=x
   IF y>ymax THEN ymax=y
   x(ixy%)=x
   y(ixy%)=y
   ixy%=ixy%+1
   menu0%=MENU(0) : IF menu0%>0 THEN mselect
WEND
LSET ca4$=MKS$(xmin) : LSET cb4$=MKS$(xmax)
PUT #1, equpt&+1
LSET ca4$=MKS$(ymin) : LSET cb4$=MKS$(ymax)
PUT #1, equpt&+2
ixy%=1
xyF%=1
RETURN

readv:
' PRINT "Reading V"
' vpt&=1+2*Nodes%
IF (TestV% AND 1) THEN
   IF iv%=1 THEN GET #1,1+vpt& : vmin=CVS(ca4$) : vmax=CVS(ca4$)
   WHILE iv%<Nodes1%
      GET #1, iv%+vpt&
      v=CVS(ca4$)
      IF v<vmin THEN vmin=v
```

```
      IF v>vmax THEN vmax=v
      v(iv%)=v : iv%=iv%+1
      menu0%=MENU(0) : IF menu0%>0 THEN mselect
    WEND
  ELSE
    WHILE iv%<Nodes1%
      v(iv%)=0 ' Stuff with 0 Potentials
      LSET ca4$=MKS$(0)
      PUT #1, iv%+vpt&
      iv%=iv%+1
      menu0%=MENU(0) : IF menu0%>0 THEN mselect
    WEND
    ' PRINT "Potentials initialized to zero"
    vmin=0 : vmax=0
    TestV%=TestV% OR 1
    GET #1, 1
    LSET ac2$=MKI$(TestV%)
    PUT #1, 1
END IF
LSET ca4$=MKS$(vmin) : LSET cb4$=MKS$(vmax)
PUT #1, equpt&+3
iv%=1
vF%=1
RETURN

readnijkl: ' Element  Ni      Nj      Nk      Nl"
' PRINT "Reading Ni, Nj, Nk, Nl"
' nijklpt&=1+3*Nodes%
WHILE inijkl%<Elements1%
  GET #1, inijkl%+nijklpt&
  Ni%(inijkl%)=CVI(aa2$)
  Nj%(inijkl%)=CVI(ab2$)
  Nk%(inijkl%)=CVI(ac2$)
  Nl%(inijkl%)=CVI(ad2$)
  inijkl%=inijkl%+1
  menu0%=MENU(0) : IF menu0%>0 THEN mselect
WEND
inijkl%=1
nijklF%=1
RETURN

readmcurd: ' Element  Mate    Curd
' PRINT "Reading Mate, Curd"
' mcurdpt&=1+3*Nodes%+Elements%
WHILE imcurd%<Elements1%
  GET #1, imcurd%+mcurdpt&
  mate%(imcurd%)=CVI(ba2$)
  Curd(imcurd%)=CVS(bc4$)
  imcurd%=imcurd%+1
  menu0%=MENU(0) : IF menu0%>0 THEN mselect
WEND
imcurd%=1
mcurdF%=1
RETURN

readu: ' Element  Perm
' PRINT "Reading u1, u2"
' upt&=1+3*Nodes%+2*Elements%
WHILE iu%<Elements1%
  GET #1, iu%+upt&
  u1(iu%)=CVS(ca4$)
  u2(iu%)=CVS(cb4$)
  iu%=iu%+1
  menu0%=MENU(0) : IF menu0%>0 THEN mselect
```

```
WEND
iu%=1
uF%=1
RETURN

Findnreg: ' Element  Region
' PRINT "Finding number of regions"
' regpt&=1+3*nodes%+3*elements%
WHILE ireg%<Elements1%
   GET #1, ireg%+regpt&
   Region%=CVI(aa2$)
   IF Region%>nreg% THEN nreg%=Region%
   ireg%=ireg%+1
   menu0%=MENU(0) :  IF menu0%>0 THEN mselect
WEND
ireg%=1
nregF%=1
RETURN

FindNabcd: '
' PRINT "Creating Triagonal element types from square types where appropriate"
' nabcdpt&=1+3*Nodes%+4*Elements%
WHILE inabcd%<Elements1%
   GET #1, inabcd%+nabcdpt&
   Na%(inabcd%)=CVI(aa2$)
   Nb%(inabcd%)=CVI(ab2$)
   Nc%(inabcd%)=CVI(ac2$)
   Nd%(inabcd%)=CVI(ad2$)
   inabcd%=inabcd%+1
   menu0%=MENU(0) :  IF menu0%>0 THEN mselect
WEND
inabcd%=1
nabcdF%=1
RETURN

FindType:
' PRINT "Splitting Elements into triangular where appropriate"
IF xyF%=0 THEN GOSUB readxy
IF nijklF%=0 THEN GOSUB readnijkl
WHILE itype%<Elements1%
   GET #1, itype%+typept&
   Type%=CVI(bb2$)
   IF Type%=0 THEN
      n1%=Ni%(i%) :  n2%=Nj%(i%) :  n3%=Nk%(i%) :  n4%=Nl%(i%)
      xi=x(n1%) :  yi=y(n1%)
      xj=x(n2%) :  yj=y(n2%)
      xk=x(n3%) :  yk=y(n3%)
      xl=x(n4%) :  yl=y(n4%)
      d1=(xk-xi)^2+(yk-yi)^2
      d2=(xl-xj)^2+(yl-yj)^2
      Type%=1
      IF d1<d2*.9999 THEN Type%=2
      LSET bb2$=MKI$(Type%)
      PUT #1, itype%+typept&
      ' PRINT i%,Type%
   END IF
   Type%(itype%)=Type%
   itype%=itype%+1
   menu0%=MENU(0) :  IF menu0%>0 THEN mselect
WEND
itype%=1
typeF%=1
RETURN
```

```
findPnode:
'SHARED xyF%,Pnode%,StartVequ&,equpt&,TestV%
IF xyF%=0 THEN GOSUB readxy
PRINT .07,.08
INPUT "Coordinates of Node to monitor during solve:  x, y   ",xn,yn
dmin=(x(1)-xn)^2+(y(1)-yn)^2 :  Pnode%=1
FOR i%=2 TO Nodes%
   d=(x(i%)-xn)^2+(y(i%)-yn)^2
   IF d<dmin THEN dmin=d :  Pnode%=i%
NEXT i%
PRINT "Will monitor Node";Pnode%

StartVequ&=equpt&+4  '       *** Potential equations start here

GET #1, StartVequ&
LSET bb2$=MKI$(Pnode%)  ' Node to print out
PUT #1, StartVequ&
TestV%=TestV% OR 2
GET #1, 1
LSET ac2$=MKI$(TestV%)
PUT #1, 1
RETURN

'
'  GRAPHIC ROUTINES FOR DEFAULT WINDOW
'
'  Define graphic coordinates of default window (quadrant 1 coordinates).
'    x1,x2,y1,y2       = coordinates, x1<x2, y1<y2
'    ABS(windowfraction) = is % of window that the coordinates x1,x2,y1,y2
'                represent.  (0.9 is a good number to use)
'  NOTE:  windowfraction<1 forces linear x-y scaling
'
SUB frame(x1,x2,y1,y2,windowfraction) STATIC
 SHARED xmultiplierscale,xshiftscale,ymultiplierscale,yshiftscale
 test=1-SGN(windowfraction)
 windowfraction=ABS(windowfraction)
 wx%=WINDOW(2)-1    ' window width
 Hy%=WINDOW(3)-1    ' window height
 x10=INT((1-windowfraction)*wx%/2)  ' SYSTEM(5)=screen width
 y10=INT((1-windowfraction)*Hy%/2)  ' SYSTEM(6)=screen height
 x20=wx%-x10
 y20=Hy%-y10
 xmultiplierscale=(x20-x10)/(x2-x1)
 ymultiplierscale=(y10-y20)/(y2-y1)
 xmult=ABS(xmultiplierscale)
 ymult=ABS(ymultiplierscale)
 IF test AND xmult>ymult THEN xmultiplierscale=ymult*SGN(xmultiplierscale)
 IF test AND ymult>xmult THEN ymultiplierscale=xmult*SGN(ymultiplierscale)
 xshiftscale=x10-xmultiplierscale*x1
 yshiftscale=y20-ymultiplierscale*y1
END SUB

'
'  Move pen to position specifed by the frame subroutine coordinate system.
'
SUB pento(x1,y1) STATIC
 SHARED xmultiplierscale,xshiftscale,ymultiplierscale,yshiftscale
 MOVETO x1*xmultiplierscale+xshiftscale,y1*ymultiplierscale+yshiftscale
END SUB

'
'  Draw pen to position specifed by the frame subroutine coordinate system.
'
SUB drawto(x2,y2) STATIC
```

```
 SHARED xmultiplierscale,xshiftscale,ymultiplierscale,yshiftscale
 LINETO x2*xmultiplierscale+xshiftscale,y2*ymultiplierscale+yshiftscale
END SUB

'
' Draw a line as specifed by the frame subroutine coordinate system.
'
SUB draw(x1,y1,x2,y2) STATIC
 SHARED xmultiplierscale,xshiftscale,ymultiplierscale,yshiftscale
 MOVETO x1*xmultiplierscale+xshiftscale,y1*ymultiplierscale+yshiftscale
 LINETO x2*xmultiplierscale+xshiftscale,y2*ymultiplierscale+yshiftscale
END SUB

' Compare two real values to roughly the first 6 digits
' 1%=0   a<>b
' 1%=-1  a=b
SUB compare(a,b,1%) STATIC
 a&=VARPTR(a) : POKEL a&, PEEKL(a&) + 128 : POKE a&+3, 0
 a&=VARPTR(b) : POKEL a&, PEEKL(a&) + 128 : POKE a&+3, 0
 1%=(a=b)
END SUB

' Inter (x1,y1,x2,y2,x,y)
' Linear interpolation 2
' Find y given x and two data points (x1,y1), (x2,y2)

SUB Inter(x1,y1,x2,y2,x,y) STATIC
 y=(x-x1)/(x2-x1)*(y2-y1)+y1
END SUB

' Intersect (xa1,ya1,xa2,ya2,xb1,yb1,xb2,yb2,x,y,test%)
' Given two lines, one passing through (xa1,ya1), (xa2,ya2)
' and the other passing through (xb1,yb1), (xb2,yb2),
' Find point of intersection, (x,y)
' Lines must have non-zero length, but may be horizontal or vertical
' Test%=0  - No intersection
' Test%=1  - One point of intersection
' Test%=2  - Infinite points of intersection
' *** Modification:  (x,y) must be on line segment 'a' for test%>0

SUB Intersect(xa1,ya1,xa2,ya2,xb1,yb1,xb2,yb2,x,y,test%) STATIC
 dxa=xa2-xa1
 dxb=xb2-xb1
 dya=ya2-ya1
 dyb=yb2-yb1
 denom=dxa*dyb-dxb*dya
 IF denom=0 THEN
    test1=dyb*(xa1-xb1)
    test2=dxb*(ya1-yb1)
    a&=VARPTR(test1) : POKEL a&, PEEKL(a&) + 128 : POKE a&+3, 0
    a&=VARPTR(test2) : POKEL a&, PEEKL(a&) + 128 : POKE a&+3, 0
    IF test1=test2 THEN test%=2 : EXIT SUB
    test%=0 : EXIT SUB
 END IF
 test%=1
 y=(dya*dyb*(xb1-xa1)+ya1*dxa*dyb-yb1*dya*dxb)/denom  ' point of intersection
 x=(dxa*dxb*(ya1-yb1)-dya*dxb*xa1+dyb*dxa*xb1)/denom
 x1=x*.9999  '***
 x2=x*1.0001  '***
 y1=y*.9999  '***
 y2=y*1.0001  '*** to end
 IF (x2<xa1 AND x2<xa2) OR  (x1>xa1 AND x1>xa2)  OR (y2<ya1 AND y2<ya2) OR  (y1>ya1 AND
y1>ya2) THEN test%=0 : EXIT SUB  '***
END SUB
```

```
SUB defnodecond STATIC
 SHARED x1,y1,x2,y2
 SHARED nuvcF%,inuvc%,dx,dy,Nodes%,xyF%,x(),y(),Nu%(),menu0%
 ' Trashing info in Nu%()
 nuvcF%=0 : inuvc%=1
 ' Assumes xyF%=1
 IF xyF%=0 THEN PRINT "ERROR in sub defnodecond:" : PRINT "x() and y() not defined" : INPUT x :
STOP
 dx=x2-x1
 dy=y2-y1
 Nu%(0)=0
 FOR i%=1 TO Nodes%
   x=x(i%)
   y=y(i%)
   test1=dy*(x-x1)+(y1-y*1.0001)*dx
   test2=dy*(x-x1)+(y1-y*.99999)*dx
   IF test1<0 AND test2<0 THEN
     ' Point A is to the right of a line going from Point 1 to Point 2
     test%=1
    ELSEIF test1>0 AND test2>0 THEN
     ' Point A is to the left of a line going from Point 1 to Point 2
     test%=4
    ELSE
     ' Point A is on the line going from Point 1 to Point 2
     test%=2
   END IF
   x1t=x1 : x2t=x2 : y1t=y1 : y2t=y2
   '  Round off so real comparison works
   a&=VARPTR(x)                  ' get variable pointer
   POKEL a&, PEEKL(a&) + 128  ' equivent to INT(a+.5)
   POKE a&+3, 0                  ' truncate last 8 bits
   a&=VARPTR(x1t) : POKEL a&, PEEKL(a&) + 128 : POKE a&+3, 0
   a&=VARPTR(x2t) : POKEL a&, PEEKL(a&) + 128 : POKE a&+3, 0
   a&=VARPTR(y)   : POKEL a&, PEEKL(a&) + 128 : POKE a&+3, 0
   a&=VARPTR(y1t) : POKEL a&, PEEKL(a&) + 128 : POKE a&+3, 0
   a&=VARPTR(y2t) : POKEL a&, PEEKL(a&) + 128 : POKE a&+3, 0
   IF x<=x1t AND x<=x2t THEN
     IF x=x1t AND x=x2t THEN
       test%=test% OR 16
      ELSE
       test%=test% OR 8
     END IF
    ELSEIF x>=x1t AND x>=x2t THEN
     test%=test% OR 32
    ELSE
     test%=test% OR 16
   END IF
   IF y<=y1t AND y<=y2t THEN
     IF y=y1t AND y=y2t THEN
       test%=test% OR 128
      ELSE
       test%=test% OR 64
     END IF
    ELSEIF y>=y1t AND y>=y2t THEN
     test%=test% OR 256
    ELSE
     test%=test% OR 128
   END IF
   Nu%(i%)=test%
   ' PRINT i%;test%;test% AND 448;test% AND 56;test% AND 7
   menu0%=MENU(0) : IF menu0%>0 THEN WINDOW 1 : EXIT SUB
 NEXT i%
 dx=ABS(dx)
```

```
 dy=ABS(dy)
END SUB


SUB getdlelem(n%,begin%,elem%,xh1,yh1,xh2,yh2,factor) STATIC
 ' Assumes this routine will be used directly after sub 'defnodecond'
 ' Variables x1,y1,x2,y2,dx,dy,Nu%() come from defnodecond
 ' Info in Nu%() and Vc() is trashed.
 ' n%=1 or 2. selects which triangle of pair to investigate
 ' start seaching elements at begin% for intersection with line x1,y1-x2,y2
 ' elem% is first element found with intersection
 ' if elem%=0 then search is finished
 ' xh1,yh1,xh2,yh2 are coordinates of the intersection
 ' normally factor=1.
 ' factor=.5 means the intersection is shared by two elements
 ' n11%,n12%,n13%,type% ' (saves having to look it up later)
 SHARED Elements%,Ni%(),Nj%(),Nk%(),Nl%(),Type%(),Vc(),plot(),x(),y()
 SHARED x1,y1,x2,y2,dx,dy,Nu%(),menu0%
 SHARED n11%,n12%,n13%,Type% ' (saves having to look it up later)
 FOR i%=begin% TO Elements%
   n1%=Ni%(i%) : n2%=Nj%(i%) : n3%=Nk%(i%) : n4%=Nl%(i%)
   Type%=Type%(i%)
   SELECT CASE Type%
    CASE 1
     IF n%=1 THEN n11%=n1% : n12%=n2% : n13%=n4% ELSE n11%=n2% : n12%=n3% :
n13%=n4%
    CASE 2
     IF n%=1 THEN n11%=n1% : n12%=n2% : n13%=n3% ELSE n11%=n1% : n12%=n3% :
n13%=n4%
    CASE ELSE  'triangle
     IF n%=1 THEN n11%=n1% : n12%=n2% : n13%=n3% ELSE n11%=0 : n12%=0 : n13%=0
   END SELECT
   nun11%=Nu%(n11%) : nun12%=Nu%(n12%) : nun13%=Nu%(n13%)
   test1%=nun11% AND nun12% AND nun13%
   'Eliminate elements outside box volume
   IF test1% AND 360 THEN nextelem ' 8 and  32 and 64 and 256
   test2%=((nun11% AND 2)=0) + ((nun12% AND 2)=0) + ((nun13% AND 2)=0)
   IF test2%=-3 THEN ' No nodes on line
     'Eliminate elements above or below line
     IF test1% AND 5 THEN nextelem ' 1 and 4
     factor=1
    ELSEIF test2%=-2 THEN ' One node on line
     'Eliminate elements above or below line
     IF (nun11% AND 2) THEN
       IF nun12% AND nun13% AND 5 THEN nextelem
     END IF
     IF (nun12% AND 2) THEN
       IF nun11% AND nun13% AND 5 THEN nextelem
     END IF
     IF (nun13% AND 2) THEN
       IF nun11% AND nun12% AND 5 THEN nextelem
     END IF
     factor=1
    ELSE ' Two nodes on line
     IF test2%=0 THEN nextelem ' Bad element
     factor=.5
   END IF
   ' Element has survived basic tests-- now look further
   xa1=x(n11%) : ya1=y(n11%)
   xa2=x(n12%) : ya2=y(n12%)
   xa3=x(n13%) : ya3=y(n13%)
   Intersect xa1,ya1,xa2,ya2,x1,y1,x2,y2,xh1,yh1,test1%
   Intersect xa2,ya2,xa3,ya3,x1,y1,x2,y2,xh2,yh2,test2%
   Intersect xa3,ya3,xa1,ya1,x1,y1,x2,y2,xh3,yh3,test3%
```

```
test%=0
IF test1%=1 THEN
   test%=test%+1
   Vc(test%)=xh1 :  Vc(test%+8)=yh1
 ELSEIF test1%=2 THEN
   test%=test%+1
   Vc(test%)=xa1 :  Vc(test%+8)=ya1
   test%=test%+1
   Vc(test%)=xa2 :  Vc(test%+8)=ya2
 ELSE
END IF
IF test2%=1 THEN
   test%=test%+1
   Vc(test%)=xh2 :  Vc(test%+8)=yh2
 ELSEIF test2%=2 THEN
   test%=test%+1
   Vc(test%)=xa2 :  Vc(test%+8)=ya2
   test%=test%+1
   Vc(test%)=xa3 :  Vc(test%+8)=ya3
 ELSE
END IF
IF test3%=1 THEN
   test%=test%+1
   Vc(test%)=xh3 :  Vc(test%+8)=yh3
 ELSEIF test3%=2 THEN
   test%=test%+1
   Vc(test%)=xa3 :  Vc(test%+8)=ya3
   test%=test%+1
   Vc(test%)=xa1 :  Vc(test%+8)=ya1
 ELSE
END IF
IF test%=1 THEN nextelem
IF dx>dy THEN
   ' Make comparisons based on x values
   xhmin=Vc(1) :  nmin%=1
   xhmax=xhmin :  nmax%=1
   FOR j%=2 TO test%
      x=Vc(j%)
      IF x<xhmin THEN xhmin=x :  nmin%=j%
      IF x>xhmax THEN xhmax=x :  nmax%=j%
   NEXT j%
   compare xhmin,xhmax,l%
   IF l% THEN nextelem ' length of line segment is zero
   IF (xhmax<=x1 AND xhmax<=x2) OR (xhmin>=x1 AND xhmin>=x2) THEN nextelem
   plot(1)=Vc(nmin%+8) :  plot(2)=Vc(nmax%+8)
   Vc(1)=xhmin :  Vc(2)=xhmax
   Vc(3)=x1 :  plot(3)=y1
   Vc(4)=x2 :  plot(4)=y2
   FOR j%=1 TO 3
      FOR k%=j%+1 TO 4
         IF Vc(j%)>Vc(k%) THEN
            x=Vc(k%) :  p=plot(k%)
            Vc(k%)=Vc(j%) :  plot(k%)=plot(j%)
            Vc(j%)=x :  plot(j%)=p
         END IF
      NEXT k%
   NEXT j%
   xh1=Vc(2) :  yh1=plot(2)
   xh2=Vc(3) :  yh2=plot(3)
   IF SGN(x2-x1)<>SGN(xh2-xh1) THEN
      ' Make sure segment is going in same direction as line of integration!
      SWAP xh1, xh2
      SWAP yh1, yh2
   END IF
```

```
    ELSE
     ' Make comparisons based on y values
     xhmin=Vc(9) : nmin%=1
     xhmax=xhmin : nmax%=1
     FOR j%=2 TO test%
       x=Vc(j%+8)
       IF x<xhmin THEN xhmin=x : nmin%=j%
       IF x>xhmax THEN xhmax=x : nmax%=j%
     NEXT j%
     compare xhmin,xhmax,l%
     IF l% THEN nextelem ' length of line segment is zero
     IF (xhmax<=y1 AND xhmax<=y2) OR (xhmin>=y1 AND xhmin>=y2) THEN nextelem
     plot(1)=Vc(nmin%) : plot(2)=Vc(nmax%)
     Vc(1)=xhmin : Vc(2)=xhmax
     Vc(3)=y1 : plot(3)=x1
     Vc(4)=y2 : plot(4)=x2
     'PRINT "%";Vc(1);Vc(2);Vc(3);Vc(4);Vc(5);Vc(6)
     FOR j%=1 TO 3
       FOR k%=j%+1 TO 4
         IF Vc(j%)>Vc(k%) THEN
           x=Vc(k%) : p=plot(k%)
           Vc(k%)=Vc(j%) : plot(k%)=plot(j%)
           Vc(j%)=x : plot(j%)=p
         END IF
       NEXT k%
     NEXT j%
     'PRINT "#";Vc(1);Vc(2);Vc(3);Vc(4);Vc(5);Vc(6)
     yh1=Vc(2) : xh1=plot(2)
     yh2=Vc(3) : xh2=plot(3)
     IF SGN(y2-y1)<>SGN(yh2-yh1) THEN
       ' Make sure segment is going in same direction as line of integration!
       SWAP xh1, xh2
       SWAP yh1, yh2
     END IF
   END IF
   elem%=i%
   EXIT SUB

   nextelem: menu0%=MENU(0) : IF menu0%>0 THEN WINDOW 1 : EXIT SUB
 NEXT i%
 elem%=-1
END SUB
```

MODEL GENERATION PROGRAM

This program creates a finite element model of two iron bars (upper and lower) with an iron stator and rotor above and below. The dimensions of the bars and air spaces and current densities are read in from a text file. A sample text file is shown below:

```
    '   Contra Rotating Demo Motor  Br=.76 Tesla, Ibar =35000
Amp
    W0=.02797          ' Width of bars
    G0=.00127          ' Gap between bar and radial symmetry
    H1=.04191          ' Height of lower bar
    H2=.032258         ' Height of upper bar
    G1=.00127          ' Gap between Iron core and lower bar
    G2=.00254          ' Gap between Iron bars
```

```
    G3=.00127          ' Gap between Iron core and upper bar
    HC=.015            ' Height of Iron core, top and bottom
    CURD1=-876419.0      ' Current density of lower bar
    CURD2=1133616.0      ' Current density of upper bar
    by=.76             ' Radial Flux density
    dn=699             ' Number of nodes/meter
```

The iron bar model is "hard wired" into the program. Dimensions and current densities can be changed, but not the structure of the model.

```
'  CREATE INPUT MESH FILE FOR RICHARD'S FE PROG 7/18/89
'  TWO IRON BARS ON TOP ONE ANOTHER WITH AXIAL CURRENT
'     AND RADIAL FLUX
'  SIDES OF BARS SEPARATED BY AIR
'     WITH SYMMETRY + CONSTANT BOUNDRY CONDITION.
'
'  Read input File, Then creates or overwrites output file named FileName$ as #1

COMMON WhoAmI$,FileName$,Printf%,menu0%,menu1%
' WhoAmI$ is Program name to return to (the shell program)
' FileName$ is the name of the Random acess file to be created by
'     this program. (The shell program needs to know this)
' Printf% is a flag indicating whether file prints are to the screen
'     or to a file. (The shell program needs to remember this)
' menu0%, menu1% points where to return to in the main program.
'     DO NOT DEFINE OR CHANGE menu0% OR menu1%.
MENU RESET

Begin: CN$=FILES$(1,"TEXT")
IF CN$="" THEN CHAIN WhoAmI$
OPEN CN$ FOR INPUT AS #3
pathmark%=1
WHILE INSTR(pathmark%,CN$,":")>0
  pathmark%=INSTR(pathmark%,CN$,":")+1
WEND
FileName$=MID$(CN$,pathmark%,pathmark%+25)+".mesh"
PRINT FileName$

changeCursor 4

TWO%=0 : TGO%=0 : TH1%=0 : TH2%=0 : TG1%=0 : TG2%=0 : TG3%=0 : THC%=0
TCURD1%=0 : TCURD2%=0 : TBy%=0 : Tdn%=0

WHILE NOT(EOF(3))
  LINE INPUT #3, A$
  NC%=INSTR(A$,"=")
  IF NC%>1 THEN
    V=VAL(MID$(A$,NC%+1))
    A$=UCASE$(MID$(A$,1,NC%-1))
    PRINT A$,V
    IF A$="WO" THEN WO=V : TWO%=1
    IF A$="GO" THEN GO=V : TGO%=1
    IF A$="H1" THEN H1=V : TH1%=1
    IF A$="H2" THEN H2=V : TH2%=1
    IF A$="G1" THEN G1=V : TG1%=1
    IF A$="G2" THEN G2=V : TG2%=1
    IF A$="G3" THEN G3=V : TG3%=1
    IF A$="HC" THEN HC=V : THC%=1
    IF A$="CURD1" THEN CURD1=V : TCURD1%=1
    IF A$="CURD2" THEN CURD2=V : TCURD2%=1
    IF A$="BY" THEN By=V :T By%=1
    IF A$="DN" THEN dn=V :T dn%=1
  END IF
```

```
WEND
CLOSE #3

TEST%=TWO%*TGO%*TH1%*TH2%*TG1%*TG2%*TG3%*THC%*TCURD1%*TCURD2%*TBy%
*Tdn%
IF TEST%=0 THEN PRINT "ERROR IN INPUT FILE  Variable Not Defined" :  INPUT x

IF WO*GO*H1*H2*G1*G2*G3*HC=0 THEN PRINT "ERROR IN INPUT FILE  Bad Variable" :
INPUT x

Vconst=By*(GO+WO+GO)
PRINT "Vconst =";Vconst

NGO=CINT(dn*GO+.5)
NWO=CINT(dn*WO+.5)
NHC=CINT(dn*HC+.5)
NG1=CINT(dn*G1+.5)
NH1=CINT(dn*H1+.5)
NG2=CINT(dn*G2+.5)
NH2=CINT(dn*H2+.5)
NG3=CINT(dn*G3+.5)

PRINT
PRINT NGO,NWO,NGO
PRINT
PRINT NHC
PRINT NG3
PRINT NH2
PRINT NG2
PRINT NH1
PRINT NG1
PRINT NHC

loop: n1=1
n2=n1+NHC
n3=n2+NG1
...select: n4=n3+NH1
n5=n4+NG2
n6=n5+NH2
while2: n7=n6+NG3
scase: n8=n7+NHC
n9=n8*NGO+1
n10=n9+NHC
n11=n10+NG1
n12=n11+NH1
n13=n12+NG2
n14=n13+NH2
n15=n14+NG3
N16=n15+NHC
n17=N16+n8*(NWO-1)+1
n18=n17+NHC
n19=n18+NG1
n20=n19+NH1
n21=n20+NG2
n22=n21+NH2
n23=n22+NG3
N24=n23+NHC
n25=N24+n8*(NGO-1)+1
n26=n25+NHC
n27=n26+NG1
n28=n27+NH1
n29=n28+NG2
n30=n29+NH2
n31=n30+NG3
```

```
N32=n31+NHC

PRINT
PRINT n8,N16,N24,N32
Finished: PRINT n7,n15,n23,n31
PRINT n6,n14,n22,n30
PRINT n5,n13,n21,n29
PRINT n4,n12,n20,n28
PRINT n3,n11,n19,n27
GoHome: PRINT n2,n10,n18,n26
PRINT n1,n9,n17,n25

n%=INSTR(CN$,":") :  NC%=0
WHILE n%>0
   error1: NC%=n%
   n%=INSTR(n%+1,CN$,":")
WEND

CLOSE #1
OPEN "O",#1,FileName$,8
CLOSE #1
OPEN "R",#1,FileName$,8
NAME FileName$ AS FileName$, "MESH"

FIELD #1, 2 AS aa2$, 2 AS ab2$, 2 AS ac2$, 2 AS ad2$
FIELD #1, 2 AS ba2$, 2 AS bb2$, 4 AS bc4$
error2: FIELD #1, 4 AS ca4$, 4 AS cb4$

Nodes%=N32
PRINT "*** NODES =";Nodes%;"***"

n&=1   ' Node #1 is stored in Record #2, Node#2 in Record#3, and so on...
dx=GO/NGO
x=-dx
FOR i%=n1 TO n9 STEP n8
   x=x+dx
   dy=HC/NHC
   y=-dy
   FOR j%=n1 TO n2
      Abort: n&=n&+1
      y=y+dy
      LSET ba2$=MKI$(n&-1)   ' Nu%
      LSET bc4$=MKS$(0)   ' Vc
      PUT #1, n&
      LSET ca4$=MKS$(x)   ' x
      LSET cb4$=MKS$(y)   ' y
      PUT #1, n&+Nodes%
   NEXT j%

   dy=G1/NG1
   FOR j%=n2+1 TO n3
      n&=n&+1
      y=y+dy
      LSET ba2$=MKI$(n&-1)   ' Nu%
      LSET bc4$=MKS$(0)   ' Vc
      PUT #1, n&
      LSET ca4$=MKS$(x)   ' x
      LSET cb4$=MKS$(y)   ' y
      PUT #1, n&+Nodes%
   NEXT j%

   dy=H1/NH1
   FOR j%=n3+1 TO n4
      n&=n&+1
```

```
      y=y+dy
      LSET ba2$=MKI$(n&-1)  'Nu%
      LSET bc4$=MKS$(0)  'Vc
      PUT #1, n&
      LSET ca4$=MKS$(x)  'x
      LSET cb4$=MKS$(y)  'y
      PUT #1, n&+Nodes%
    NEXT j%

    dy=G2/NG2
    FOR j%=n4+1 TO n5
      n&=n&+1
      y=y+dy
      LSET ba2$=MKI$(n&-1)  'Nu%
      LSET bc4$=MKS$(0)  'Vc
      PUT #1, n&
      LSET ca4$=MKS$(x)  'x
      LSET cb4$=MKS$(y)  'y
      PUT #1, n&+Nodes%
    NEXT j%

    dy=H2/NH2
    FOR j%=n5+1 TO n6
      n&=n&+1
      y=y+dy
      LSET ba2$=MKI$(n&-1)  'Nu%
      LSET bc4$=MKS$(0)  'Vc
      PUT #1, n&
      LSET ca4$=MKS$(x)  'x
      LSET cb4$=MKS$(y)  'y
      PUT #1, n&+Nodes%
    NEXT j%

    dy=G3/NG3
    FOR j%=n6+1 TO n7
      n&=n&+1
      y=y+dy
      LSET ba2$=MKI$(n&-1)  'Nu%
      LSET bc4$=MKS$(0)  'Vc
      PUT #1, n&
      LSET ca4$=MKS$(x)  'x
      LSET cb4$=MKS$(y)  'y
      PUT #1, n&+Nodes%
    NEXT j%

    dy=HC/NHC
    FOR j%=n7+1 TO n8
      n&=n&+1
      y=y+dy
      LSET ba2$=MKI$(n&-1)  'Nu%
      LSET bc4$=MKS$(0)  'Vc
      PUT #1, n&
      LSET ca4$=MKS$(x)  'x
      LSET cb4$=MKS$(y)  'y
      PUT #1, n&+Nodes%
    NEXT j%

  NEXT i%

  dx=WO/NWO
  FOR i%=n9+n8 TO n17 STEP n8
    x=x+dx
    dy=HC/NHC
    y=-dy
```

```
FOR j%=n1 TO n2
  n&=n&+1
  y=y+dy
  LSET ba2$=MKI$(n&-1)  'Nu%
  LSET bc4$=MKS$(0)  'Vc
  PUT #1, n&
  LSET ca4$=MKS$(x)  'x
  LSET cb4$=MKS$(y)  'y
  PUT #1, n&+Nodes%
NEXT j%

dy=G1/NG1
FOR j%=n2+1 TO n3
  n&=n&+1
  y=y+dy
  LSET ba2$=MKI$(n&-1)  'Nu%
  LSET bc4$=MKS$(0)  'Vc
  PUT #1, n&
  LSET ca4$=MKS$(x)  'x
  LSET cb4$=MKS$(y)  'y
  PUT #1, n&+Nodes%
NEXT j%

dy=H1/NH1
FOR j%=n3+1 TO n4
  n&=n&+1
  y=y+dy
  LSET ba2$=MKI$(n&-1)  'Nu%
  LSET bc4$=MKS$(0)  'Vc
  PUT #1, n&
  LSET ca4$=MKS$(x)  'x
  LSET cb4$=MKS$(y)  'y
  PUT #1, n&+Nodes%
NEXT j%

dy=G2/NG2
FOR j%=n4+1 TO n5
  n&=n&+1
  y=y+dy
  LSET ba2$=MKI$(n&-1)  'Nu%
  LSET bc4$=MKS$(0)  'Vc
  PUT #1, n&
  LSET ca4$=MKS$(x)  'x
  LSET cb4$=MKS$(y)  'y
  PUT #1, n&+Nodes%
NEXT j%

dy=H2/NH2
FOR j%=n5+1 TO n6
  n&=n&+1
  y=y+dy
  LSET ba2$=MKI$(n&-1)  'Nu%
  LSET bc4$=MKS$(0)  'Vc
  PUT #1, n&
  LSET ca4$=MKS$(x)  'x
  LSET cb4$=MKS$(y)  'y
  PUT #1, n&+Nodes%
NEXT j%

dy=G3/NG3
FOR j%=n6+1 TO n7
  n&=n&+1
  y=y+dy
  LSET ba2$=MKI$(n&-1)  'Nu%
```

```
      LSET bc4$=MKS$(0)   'Vc
      PUT #1, n&
      LSET ca4$=MKS$(x)   'x
      LSET cb4$=MKS$(y)   'y
      PUT #1, n&+Nodes%
    NEXT j%

    dy=HC/NHC
    FOR j%=n7+1 TO n8
      n&=n&+1
      y=y+dy
      LSET ba2$=MKI$(n&-1)   'Nu%
      LSET bc4$=MKS$(0)   'Vc
      PUT #1, n&
      LSET ca4$=MKS$(x)   'x
      LSET cb4$=MKS$(y)   'y
      PUT #1, n&+Nodes%
    NEXT j%

  NEXT i%

  dx=GO/NGO
  FOR i%=n17+n8 TO n25-n8 STEP n8
    x=x+dx
    dy=HC/NHC
    y=-dy
    FOR j%=n1 TO n2
      n&=n&+1
      y=y+dy
      LSET ba2$=MKI$(n&-1)   'Nu%
      LSET bc4$=MKS$(0)   'Vc
      PUT #1, n&
      LSET ca4$=MKS$(x)   'x
      LSET cb4$=MKS$(y)   'y
      PUT #1, n&+Nodes%
    NEXT j%

    dy=G1/NG1
    FOR j%=n2+1 TO n3
      n&=n&+1
      y=y+dy
      LSET ba2$=MKI$(n&-1)   'Nu%
      LSET bc4$=MKS$(0)   'Vc
      PUT #1, n&
      LSET ca4$=MKS$(x)   'x
      LSET cb4$=MKS$(y)   'y
      PUT #1, n&+Nodes%
    NEXT j%

    dy=H1/NH1
    FOR j%=n3+1 TO n4
      n&=n&+1
      y=y+dy
      LSET ba2$=MKI$(n&-1)   'Nu%
      LSET bc4$=MKS$(0)   'Vc
      PUT #1, n&
      LSET ca4$=MKS$(x)   'x
      LSET cb4$=MKS$(y)   'y
      PUT #1, n&+Nodes%
    NEXT j%

    dy=G2/NG2
    FOR j%=n4+1 TO n5
      n&=n&+1
```

```
    y=y+dy
    LSET ba2$=MKI$(n&-1)  'Nu%
    LSET bc4$=MKS$(0)  'Vc
    PUT #1, n&
    LSET ca4$=MKS$(x)  'x
    LSET cb4$=MKS$(y)  'y
    PUT #1, n&+Nodes%
  NEXT j%

  dy=H2/NH2
  FOR j%=n5+1 TO n6
    n&=n&+1
    y=y+dy
    LSET ba2$=MKI$(n&-1)  'Nu%
    LSET bc4$=MKS$(0)  'Vc
    PUT #1, n&
    LSET ca4$=MKS$(x)  'x
    LSET cb4$=MKS$(y)  'y
    PUT #1, n&+Nodes%
  NEXT j%

  dy=G3/NG3
  FOR j%=n6+1 TO n7
    n&=n&+1
    y=y+dy
    LSET ba2$=MKI$(n&-1)  'Nu%
    LSET bc4$=MKS$(0)  'Vc
    PUT #1, n&
    LSET ca4$=MKS$(x)  'x
    LSET cb4$=MKS$(y)  'y
    PUT #1, n&+Nodes%
  NEXT j%

  dy=HC/NHC
  FOR j%=n7+1 TO n8
    n&=n&+1
    y=y+dy
    LSET ba2$=MKI$(n&-1)  'Nu%
    LSET bc4$=MKS$(0)  'Vc
    PUT #1, n&
    LSET ca4$=MKS$(x)  'x
    LSET cb4$=MKS$(y)  'y
    PUT #1, n&+Nodes%
  NEXT j%

NEXT i%

i%=n25
nn&=n&
x=x+GO/NGO
dy=HC/NHC
y=-dy
FOR j%=n1 TO n2
  n&=n&+1
  y=y+dy
  LSET ba2$=MKI$(n&-nn&)  'Nu%
  LSET bc4$=MKS$(Vconst)  'Vc
  PUT #1, n&
  LSET ca4$=MKS$(x)  'x
  LSET cb4$=MKS$(y)  'y
  PUT #1, n&+Nodes%
NEXT j%

dy=G1/NG1
```

```
FOR j%=n2+1 TO n3
  n&=n&+1
  y=y+dy
  LSET ba2$=MKI$(n&-nn&)   'Nu%
  LSET bc4$=MKS$(Vconst)   'Vc
  PUT #1, n&
  LSET ca4$=MKS$(x)  'x
  LSET cb4$=MKS$(y)  'y
  PUT #1, n&+Nodes%
NEXT j%

dy=H1/NH1
FOR j%=n3+1 TO n4
  n&=n&+1
  y=y+dy
  LSET ba2$=MKI$(n&-nn&)   'Nu%
  LSET bc4$=MKS$(Vconst)   'Vc
  PUT #1, n&
  LSET ca4$=MKS$(x)  'x
  LSET cb4$=MKS$(y)  'y
  PUT #1, n&+Nodes%
NEXT j%

dy=G2/NG2
FOR j%=n4+1 TO n5
  n&=n&+1
  y=y+dy
  LSET ba2$=MKI$(n&-nn&)   'Nu%
  LSET bc4$=MKS$(Vconst)   'Vc
  PUT #1, n&
  LSET ca4$=MKS$(x)  'x
  LSET cb4$=MKS$(y)  'y
  PUT #1, n&+Nodes%
NEXT j%

dy=H2/NH2
FOR j%=n5+1 TO n6
  n&=n&+1
  y=y+dy
  LSET ba2$=MKI$(n&-nn&)   'Nu%
  LSET bc4$=MKS$(Vconst)   'Vc
  PUT #1, n&
  LSET ca4$=MKS$(x)  'x
  LSET cb4$=MKS$(y)  'y
  PUT #1, n&+Nodes%
NEXT j%

dy=G3/NG3
FOR j%=n6+1 TO n7
  n&=n&+1
  y=y+dy
  LSET ba2$=MKI$(n&-nn&)   'Nu%
  LSET bc4$=MKS$(Vconst)   'Vc
  PUT #1, n&
  LSET ca4$=MKS$(x)  'x
  LSET cb4$=MKS$(y)  'y
  PUT #1, n&+Nodes%
NEXT j%

dy=HC/NHC
FOR j%=n7+1 TO n8
  n&=n&+1
  y=y+dy
  LSET ba2$=MKI$(n&-nn&)   'Nu%
```

```
    LSET bc4$=MKS$(Vconst)   'Vc
    PUT #1, n&
    LSET ca4$=MKS$(x)  'x
    LSET cb4$=MKS$(y)  'y
    PUT #1, n&+Nodes%
NEXT j%

Elements%=CINT((NGO+NWO+NGO)*(NHC+NG1+NH1+NG2+NH2+NG3+NHC))
PRINT "***ELEMENTS =";Elements%;"***"

Elements2%=2*Elements%
Elements3%=3*Elements%
n&=1+3*Nodes%

NLL=n11 : NUL=n12 : NLR=n19 : mate%=3 : curd=CURD1 : u=.001  'Lower Bar
nreg%=1
GOSUB writelement

NLL=n13 : NUL=n14 : NLR=n21 : curd=CURD2  'Upper Bar
nreg%=2
GOSUB writelement

NLL=n1 : NUL=n2 : NLR=n25 : curd=0  'Lower Core
nreg%=3
GOSUB writelement

NLL=n7 : NUL=n8 : NLR=n31  'Upper Core
nreg%=4
GOSUB writelement

NLL=n2 : NUL=n3 : NLR=n26 : mate%=0 : u=1  'Lower Air Gap
nreg%=5
GOSUB writelement

NLL=n4 : NUL=n5 : NLR=n28  'Mid Air Gap
nreg%=6
GOSUB writelement

NLL=n6 : NUL=n7 : NLR=n31  'Upper Air Gap
nreg%=7
GOSUB writelement

NLL=n3 : NUL=n4 : NLR=n11  'Lower Left Air Gap
nreg%=8
GOSUB writelement

NLL=n5 : NUL=n6 : NLR=n13  'Upper Left Air Gap
nreg%=9
GOSUB writelement

NLL=n19 : NUL=n20 : NLR=n27  'Lower Right Air Gap
nreg%=10
GOSUB writelement

NLL=n21 : NUL=n22 : NLR=n29  'Upper Right Air Gap
nreg%=11
GOSUB writelement

IF n&-(1+3*Nodes%)<>Elements% THEN PRINT "ERROR IN ELEMENT GENERATION" : INPUT x

LSET aa2$=MKI$(Nodes%)
LSET ab2$=MKI$(Elements%)
LSET ac2$=MKI$(0)  'TestV%
LSET ad2$=MKI$(0)  'TestE%
```

```
PUT #1, 1
' Record #1 Stores number of elements and nodes, and flags indicating
' that Potential and Equation data have not yet been computed.

CHAIN WhoAmI$

writelement:
FOR i%=NLL TO NLR-n8 STEP n8
  FOR j%=0 TO NUL-NLL-1
    k%=i%+j%
    n&=n&+1
    LSET aa2$=MKI$(k%)       'Ni%
    LSET ab2$=MKI$(k%+1)     'Nj%
    LSET ac2$=MKI$(k%+1+n8)  'Nk%
    Hagain: LSET ad2$=MKI$(k%+n8)    'Nl%
    PUT #1, n&
    '  PRINT n&;k%;k%+1;k%+1+n8;k%+n8;mate%;curd;u
    LSET ba2$=MKI$(mate%)    'mate%
    LSET bb2$=MKI$(0)        'Type% (0=split into best triangles)
    LSET bc4$=MKS$(curd)     'curd
    PUT #1, n&+Elements%
    LSET ca4$=MKS$(u)        'u1=1/μ
    LSET cb4$=MKS$(u)        'u2=1/μ
    PUT #1, n&+Elements2%
    LSET aa2$=MKI$(nreg%)    'Region&
    LSET ab2$=MKI$(0)        'Blank
    LSET ac2$=MKI$(0)        'Blank
    LSET ad2$=MKI$(0)        'Blank
    PUT #1, n&+Elements3%
  NEXT j%
NEXT i%
RETURN
```

FIND NODE AND ELEMENT CONNECTIONS PROGRAM

This program searches the geometry part of the file to find out which elements are adjacent to each other and which nodes are connected to any given node so that they will be used in the equation for that node. The element connection data is needed by the main program to plot mesh geometry, flux plots and material outlines. Node connection data is needed by the equation writer program.

```
' Find Element Connections in 2D  9/15/89.
' Called by Finite Element Solver Program

COMMON WhoAmI$,FileName$,Printf%,menu0%,menu1%
'CLEAR ,FRE(-1)+FRE(0)-150000&  ' Leave 150K for program heap and stack

WINDOW 2,FileName$
OPTION BASE 0
' Limitation:  16000 Nodes,  12 elements phyically connected to any given node
' Conn%(x,0) holds number of elements specified for x
' DIM Conn%(16000,12)
DIM Conn%(2000,12),nelem%(4),N%(2000,3)

MENU RESET
MENU 1,0,1,"Status"
MENU 1,1,1,"Return to Main Program without saving results"
MENU 2,0,1,"          "
MENU 3,0,1,"* Finding Node Connections *"

IF FileName$="" THEN FileName$=FILES$(1,"TEXTMESH") : OPEN "R",#1,FileName$,8
```

```
U0=.000001256637#

FIELD #1, 2 AS aa2$, 2 AS ab2$, 2 AS ac2$, 2 AS ad2$
FIELD #1, 2 AS ba2$, 2 AS bb2$, 4 AS bc4$
FIELD #1, 4 AS ca4$, 4 AS cb4$

GET #1, 1
Nodes%=CVI(aa2$) : Elements%=CVI(ab2$)
TestV%=CVI(ac2$) : TestE%=CVI(ad2$)

IF menu0%<>2 AND (TestE%=1 OR TestE%=3) THEN GoHome

WINDOW 2,"Finding Nodes connected to Element:",(80,40)-(400,70),1

' Element  Ni      Nj      Nk      Nl"
nijklpt&=1+3*Nodes%
GET #1, nijklpt&
FOR inijkl%=1 TO Elements%
   PRINT inijkl%
   GET #1
   nelem%(0)=CVI(aa2$)
   nelem%(1)=CVI(ab2$)
   nelem%(2)=CVI(ac2$)
   nelem%(3)=CVI(ad2$)
   N%(inijkl%,0)=nelem%(0)
   N%(inijkl%,1)=nelem%(1)
   N%(inijkl%,2)=nelem%(2)
   N%(inijkl%,3)=nelem%(3)
   FOR j%=0 TO 3
      Ni%=nelem%(j%)
      IF Ni%>0 THEN
         numi%=Conn%(Ni%,0)
         IF numi%=12 THEN error1
         numi%=numi%+1
         Conn%(Ni%,0)=numi%
         Conn%(Ni%,numi%)=inijkl%
      END IF
   NEXT j%
   IF MENU(0) THEN Abort
NEXT inijkl%

WINDOW 2,"Finding Elements connected to Element:",(80,40)-(400,70),1
nabcdpt&=1+3*Nodes%+4*Elements%
GET #1, nabcdpt&
FOR i%=1 TO Elements%
   PRINT i%
   loop: LSET aa2$=MKI$(0)
   LSET ab2$=MKI$(0)
   LSET ac2$=MKI$(0)
   mselect: LSET ad2$=MKI$(0)
   nelem%(0)=N%(i%,0)
   nelem%(1)=N%(i%,1)
   while2: nelem%(2)=N%(i%,2)
   scase: nelem%(3)=N%(i%,3)
   nelem%(4)=nelem%(0)
   jj%=3
   IF nelem%(3)=0 THEN nelem%(3)=nelem%(0) : jj%=2  ' Triangular Element
   FOR j%=0 TO jj%
      m%=0
      Ni%=nelem%(j%)
      Nj%=nelem%(j%+1)
      FOR k%=1 TO Conn%(Ni%,0)
         Nk%=Conn%(Ni%,k%)
         IF Nk%<>i% THEN
```

```
          FOR l%=0 TO 3
            IF N%(Nk%,l%)=Nj% THEN
              m%=m%+1
              IF m%=2 THEN error2
              IF j%=0 THEN LSET aa2$=MKI$(Nk%)
              IF j%=1 THEN LSET ab2$=MKI$(Nk%)
              IF j%=2 THEN LSET ac2$=MKI$(Nk%)
              IF j%=3 THEN LSET ad2$=MKI$(Nk%)
            END IF
          NEXT l%
        END IF
      NEXT k%
      IF MENU(0) THEN Abort
    NEXT j%
    PUT #1
NEXT i%

Finished:
GET #1, 1
IF TestE%=0 THEN LSET ad2$=MKI$(1) ELSE LSET ad2$=MKI$(3)
PUT #1, 1

GoHome:
WINDOW CLOSE 2
WINDOW 1
CHAIN WhoAmI$

error1:
WINDOW CLOSE 2
WINDOW 1
PRINT "Error in Mesh generation!"
PRINT "More that 12 elements physically connected to node";Ni%
PRINT
FOR i%=1 TO 12
  PRINT Conn%(Ni%,i%)
NEXT i%
INPUT x
GOTO Abort

error2:
WINDOW CLOSE 2
WINDOW 1
PRINT "Mesh generation error!"
PRINT  "Segment from";Ni%;" to";Nj%;" in element";i%;
PRINT " is bordered by two elements:  (only one allowed)"
IF j%=0 THEN PRINT CVI(aa2$);" and";Nk%
IF j%=1 THEN PRINT CVI(ab2$);" and";Nk%
IF j%=2 THEN PRINT CVI(ac2$);" and";Nk%
IF j%=3 THEN PRINT CVI(ad2$);" and";Nk%
INPUT x
GOTO Abort

Abort:
menu0%=0
GOTO GoHome
```

WRITE EQUATIONS PROGRAM

This program takes geometry, current, and boundary condition data from the file and generates the equations needed for calculate a solution. These equations are stored in the file in a compact form to be used by the solver.

```
' Write Equations in 2D  9/15/89.
' Called by Finite Element Shell Program

COMMON WhoAmI$,FileName$,Printf%,menu0%,menu1%
'CLEAR ,FRE(-1)+FRE(0)-150000&  ' Leave 150K for program heap and stack

WINDOW 1,FileName$
OPTION BASE 0
DIM v&(2000),pnt&(8000),elem%(8000)
DIM Nu%(2000),Vc(2000),x(2000),y(2000)
DIM Ni%(2000,3),u1(2000)
DIM mate%(2000),Type%(2000),Curd(2000)
imax%=2000  ' Maximum number of nodes SET EQUAL TO DIMENSIONED !

MENU RESET
Begin: MENU 1,0,1,"Status"
MENU 1,1,1,"Return to Main Program without saving results"
MENU 2,0,1,"                "
MENU 3,0,1,"* Writing Equations *"

IF FileName$="" THEN FileName$=FILES$(1,"TEXTMESH") :  OPEN "R",#1,FileName$,8 :
menu0%=2
U0=.000001256637#

FIELD #1, 2 AS aa2$, 2 AS ab2$, 2 AS ac2$, 2 AS ad2$
FIELD #1, 2 AS ba2$, 2 AS bb2$, 4 AS bc4$
FIELD #1, 4 AS ca4$, 4 AS cb4$

OPEN "FE.scrap" FOR OUTPUT AS #3
CLOSE #3
OPEN "FE.scrap" AS #3 LEN=12
FIELD #3, 2 AS da2$, 2 AS db2$, 4 AS dc4$, 4 AS dd4$

WINDOW 2,"Reading File Information",(80,40)-(400,70),1
'WINDOW 2,"Reading File Information",(10,40)-(500,335),1
GET #1, 1
Nodes%=CVI(aa2$) :  Elements%=CVI(ab2$)
TestV%=CVI(ac2$) :  TestE%=CVI(ad2$)

IF menu0%<>2 AND (TestE%=2 OR TestE%=3) THEN GoHome

xypt&=1+Nodes%
nijklpt&=1+3*Nodes%
mcurdpt&=1+3*Nodes%+Elements%
typept&=1+3*Nodes%+Elements%
upt&=1+3*Nodes%+2*Elements%
equpt&=2+3*Nodes%+5*Elements%

StartVequ&=equpt&+4  '       *** Potential equations start here

' PRINT "Reading Nu, Vc"
GET #1, 1
FOR inuvc%=1 TO Nodes%
   GET #1
   Nu%(inuvc%)=CVI(ba2$)
   Vc(inuvc%)=CVS(bc4$)
   IF MENU(0) THEN Abort
NEXT inuvc%

' PRINT "Reading X, Y"
GET #1, xypt&
FOR ixy%=1 TO Nodes%
   GET #1
```

```
    x(ixy%)=CVS(ca4$)
    y(ixy%)=CVS(cb4$)
    IF MENU(0) THEN Abort
NEXT ixy%

' print "Reading Nijkl"
GET #1, nijklpt&
FOR inijkl%=1 TO Elements%
    GET #1
    Ni%(inijkl%,0)=CVI(aa2$)
    Ni%(inijkl%,1)=CVI(ab2$)
    Ni%(inijkl%,2)=CVI(ac2$)
    Ni%(inijkl%,3)=CVI(ad2$)
    IF MENU(0) THEN Abort
NEXT inijkl%

loop:
' print "Reading Mate%, Type%, and Curd"
' Assumes xy and nijkl have been read in.
mselect: GET #1, mcurdpt&
FOR imcurd%=1 TO Elements%
    GET #1
    while2: mate%(imcurd%)=CVI(ba2$)
    scase: Type%(imcurd%)=CVI(bb2$)
    Curd(imcurd%)=CVS(bc4$)
    IF Type%(itype%)=0 THEN
      n1%=Ni%(i%,0) : n2%=Ni%(i%,1) : n3%=Ni%(i%,2) : n4%=Ni%(i%,3)
      xi=x(n1%) : yi=y(n1%)
      xj=x(n2%) : yj=y(n2%)
      xk=x(n3%) : yk=y(n3%)
      xl=x(n4%) : yl=y(n4%)
      d1=(xk-xi)^2+(yk-yi)^2
      d2=(xl-xj)^2+(yl-yj)^2
      ' Type%=1
      ' IF d1<d2*.9999 THEN Type%=2
      Type%=2
      IF d1>d2*1.0001 THEN Type%=1
      LSET bb2$=MKI$(Type%)
      PUT #1, imcurd%+mcurdpt&
      Type%(imcurd%)=Type%
    END IF
    IF MENU(0) THEN Abort
NEXT imcurd%

' Reading Permeability
GET #1, upt&
FOR i%=1 TO Elements%
    GET #1
    ' u() only used for linear elements (mate=0,1,2)
    ' u1<>u2 only if nonlinear element (mate>2)
    u1(i%)=CVS(ca4$)
    Finished: u2=CVS(cb4$)
    IF u1(i%)<>u2 AND mate%(i%)<3 THEN PRINT "ERROR IN MESH DATA"
    IF MENU(0) THEN Abort
NEXT i%

GoHome:
' Nu%(i)=i   : Node is independent
' Nu%(i)=j   : V(j) = V(i) + Vc(i)
' Nu%(i)=-n  : NOT YET IMPLIMENTED !!  V(n) = V(i) + Kn,  Kn to be determined by solver
(Vc(i)=Kn ref #)
' Nu%()=0    : means node has a set potential.
'   Note:  In the Solver u(0)=1 and v(0)=0
```

```
error1:
' Eliminate indirect relationships
' In the future, this could be very useful for eliminating duplicate nodes.
WINDOW 2,"Investigating Node relationships"
FOR Node%=1 TO Nodes%
  test%=0
  j%=Nu%(Node%) :  Vc=Vc(Node%)
  WHILE j%<>Nu%(j%) AND Nu%(j%)>0
    test%=1
    Vc=Vc+Vc(j%)
    j%=Nu%(j%)
    error2: IF MENU(0) THEN Abort
  WEND
  IF test% THEN Nu%(Node%)=j% :  Vc(Node%)=Vc
  ' PRINT "combine";Nu%(Node%);Vc(Node%)
  IF MENU(0) THEN Abort
NEXT Node%

' Link elements mathematically related

WINDOW 2,"Investigating Element relationships"
vpt&=0
FOR i%=1 TO Elements%
  FOR j%=0 TO 3
    Abort: n1%=ABS(Nu%(Ni%(i%,j%)))
    IF n1%>0 THEN
      vpt&=vpt&+1
      IF v&(n1%)=0 THEN spt&=vpt& ELSE spt&=pnt&(v&(n1%))
      pnt&(v&(n1%))=vpt&
      v&(n1%)=vpt&
      pnt&(vpt&)=spt&
      elem%(vpt&)=i%
    END IF
  NEXT j%
  IF MENU(0) THEN Abort
NEXT i%


' First write nodes with set potential.
nloop: ' These single line equations have the form V(Node%)=Vc(Node%)


WINDOW 2,"Finding Nodes with set potential"
GET #1, StartVequ&  ' Start of Node equations
inodes%=0  ' Number of equations for set potential nodes
FOR Node%=1 TO Nodes%  ' First write nodes with set potential.
  IF Nu%(Node%)=0 THEN
    LSET ba2$=MKI$(Node%)
    LSET bb2$=MKI$(0)  ' Blank
    LSET bc4$=MKS$(Vc(Node%))
    PUT #1
    inodes%=inodes%+1
    v&(Node%)=0
    ' PRINT "V("+MID$(STR$(Node%),2)+") =";vc(Node%)
    PRINT Node%
  END IF
  IF MENU(0) THEN Abort
NEXT Node%
StartDequ&=LOC(1)+1&  ' Start of Dependent equations
GET #1, StartVequ&
LSET ba2$=MKI$(inodes%)
' LSET bb2$=MKI$(Pnode%)  ' Node to print out  *Set in Main Program*
LSET bc4$=MKL$(StartDequ&)
PUT #1, StartVequ&
```

```
' Next write dependent node equations.
' These single line equations have the form V(Node%)=V(Nu%(Node%))+Vc(Node%)

WINDOW 2,"Finding Dependent Node equations"
GET #1, StartDequ&  ' Start of dependent node equations
dnodes%=0  ' Number of equations for dependent potential nodes
FOR Node%=1 TO Nodes%
  IF Nu%(Node%)>0 AND Nu%(Node%)<>Node% THEN
    LSET ba2$=MKI$(Node%)
    LSET bb2$=MKI$(ABS(Nu%(Node%)))
    LSET bc4$=MKS$(Vc(Node%))
    PUT #1
    dnodes%=dnodes%+1
    v&(Node%)=0
    ' PRINT "V("+MID$(STR$(Node%),2)+") = V("+MID$(STR$(ABS(Nu%(Node%))),2)+")
+";vc(Node%)
    PRINT Node%
  END IF
  IF MENU(0) THEN Abort
NEXT Node%


' Write Independent Node equations

GoHome:
currentnode&=1
WINDOW 2,"Formulating Equation for Node"
FOR Node%=1 TO Nodes%
  Finished: c1=0 : c2=0 : Nline%=0
  vpt&=v&(Node%)
  IF vpt&>0 THEN
    GET #3, currentnode&
    PRINT Node%
    sptp&=0 : spt&=pnt&(vpt&)
    WHILE spt&>sptp&
      Element%=elem%(spt&)
      n1%=Ni%(Element%,0) : n2%=Ni%(Element%,1) : n3%=Ni%(Element%,2) :
n4%=Ni%(Element%,3)
      Ja=Curd(Element%)
      SELECT CASE Type%(Element%)
       CASE 1
        n1i%=n1% : n1j%=n2% : n1k%=n4%
        n2i%=n2% : n2j%=n3% : n2k%=n4%
        IF Nu%(n1%)=Node% THEN
          test1%=1 : test2%=0 : Vc=Vc(n1%)
        ELSEIF Nu%(n2%)=Node% THEN
          test1%=2 : test2%=1 : Vc=Vc(n2%)
        ELSEIF Nu%(n3%)=Node% THEN
          test1%=0 : test2%=2 : Vc=Vc(n3%)
        ELSE
          test1%=3 : test2%=3 : Vc=Vc(n4%)
        END IF
       CASE 2
        n1i%=n1% : n1j%=n2% : n1k%=n3%
        n2i%=n1% : n2j%=n3% : n2k%=n4%
        IF Nu%(n1%)=Node% THEN
          test1%=1 : test2%=1 : Vc=Vc(n1%)
        ELSEIF Nu%(n2%)=Node% THEN
          test1%=2 : test2%=0 : Vc=Vc(n2%)
        ELSEIF Nu%(n3%)=Node% THEN
          test1%=3 : test2%=2 : Vc=Vc(n3%)
        ELSE
```

```
    test1%=0 : test2%=3 : Vc=Vc(n4%)
  END IF
 CASE 0          'Triangle- Only one will produce nonzero area.
  n1i%=n1% : n1j%=n2% : n1k%=n3%
  n2i%=n2% : n2j%=n3% : n2k%=n4%
  IF Nu%(n1%)=Node% THEN
    test1%=1 : test2%=0 : Vc=Vc(n1%)
   ELSEIF Nu%(n2%)=Node% THEN
    test1%=2 : test2%=1 : Vc=Vc(n2%)
   ELSEIF Nu%(n3%)=Node% THEN
    test1%=3 : test2%=2 : Vc=Vc(n3%)
   ELSE
    test1%=0 : test2%=3 : Vc=Vc(n4%)
  END IF
 CASE ELSE
  PRINT "Error in element types...." : INPUT x : STOP ' This shouldn't happen !
END SELECT
x1i=x(n1i%) : y1i=y(n1i%)
x1j=x(n1j%) : y1j=y(n1j%)
x1k=x(n1k%) : y1k=y(n1k%)
a14=((x1i-x1j)*(y1i-y1k)+(y1i-y1j)*(x1k-x1i)) ' 2*Area  positive direction.
'    The direction of a14 does not matter. A positive direction produces
'    a positive Kii if the element defined in a counter-clockwise direction.
IF a14<>0 THEN
  b1i=y1j-y1k : c1i=x1k-x1j
  b1j=y1k-y1i : c1j=x1i-x1k
  b1k=y1i-y1j : c1k=x1j-x1i
  k1ii=(b1i*b1i+c1i*c1i)/a14
  K1ij=(b1i*b1j+c1i*c1j)/a14
  K1ik=(b1i*b1k+c1i*c1k)/a14
  k1jj=(b1j*b1j+c1j*c1j)/a14
  K1jk=(b1j*b1k+c1j*c1k)/a14
  k1kk=(b1k*b1k+c1k*c1k)/a14
  d=b1i*b1i+c1i*c1i+b1j*b1j+c1j*c1j+b1k*b1k+c1k*c1k
  e1i=U0*Ja*a14*(b1i*b1i+c1i*c1i)/d
  e1j=U0*Ja*a14*(b1j*b1j+c1j*c1j)/d
  e1k=U0*Ja*a14*(b1k*b1k+c1k*c1k)/d
 ELSE
  k1ii=0 : K1ij=0 : K1ik=0
  k1jj=0 : K1jk=0
  k1kk=0
  e1i=0 : e1j=0 : e1k=0
END IF
IF MENU(0) THEN Abort
x2i=x(n2i%) : y2i=y(n2i%)
x2j=x(n2j%) : y2j=y(n2j%)
x2k=x(n2k%) : y2k=y(n2k%)
a24=((x2i-x2j)*(y2i-y2k)+(y2i-y2j)*(x2k-x2i)) ' 2*Area  positive direction.
IF a24<>0 THEN
  b2i=y2j-y2k : c2i=x2k-x2j
  b2j=y2k-y2i : c2j=x2i-x2k
  b2k=y2i-y2j : c2k=x2j-x2i
  k2ii=(b2i*b2i+c2i*c2i)/a24
  K2ij=(b2i*b2j+c2i*c2j)/a24
  K2ik=(b2i*b2k+c2i*c2k)/a24
  k2jj=(b2j*b2j+c2j*c2j)/a24
  K2jk=(b2j*b2k+c2j*c2k)/a24
  k2kk=(b2k*b2k+c2k*c2k)/a24
  d=b2i*b2i+c2i*c2i+b2j*b2j+c2j*c2j+b2k*b2k+c2k*c2k
  e2i=U0*Ja*a24*(b2i*b2i+c2i*c2i)/d
  e2j=U0*Ja*a24*(b2j*b2j+c2j*c2j)/d
  e2k=U0*Ja*a24*(b2k*b2k+c2k*c2k)/d
 ELSE
  k2ii=0 : K2ij=0 : K2ik=0
```

```
    k2jj=0 : K2jk=0
    k2kk=0
    e2i=0 : e2j=0 : e2k=0
  END IF
  k11=0 : K12=0 : k21=0 : k22=0
  IF test1%=1 THEN
    c1=c1+e1i
    k11=K1ij : K12=K1ik
    n11%=n1j% : n12%=n1k%
  END IF
  IF test1%=2 THEN
    c1=c1+e1j
    k11=K1ij : K12=K1jk
    n11%=n1i% : n12%=n1k%
  END IF
  IF test1%=3 THEN
    c1=c1+e1k
    k11=K1ik : K12=K1jk
    n11%=n1i% : n12%=n1j%
  END IF
  IF test2%=1 THEN
    c2=c2+e2i
    k21=K2ij : k22=K2ik
    n21%=n2j% : n22%=n2k%
  END IF
  IF test2%=2 THEN
    c2=c2+e2j
    k21=K2ij : k22=K2jk
    n21%=n2i% : n22%=n2k%
  END IF
  IF test2%=3 THEN
    c2=c2+e2k
    k21=K2ik : k22=K2jk
    n21%=n2i% : n22%=n2j%
  END IF

  IF mate%(Element%)<3 THEN
    ' PRINT "$";Element%;mate%(Element%);u1(Element%)
    k11=k11*u1(Element%)
    K12=K12*u1(Element%)
    k21=k21*u1(Element%)
    k22=k22*u1(Element%)
    Element%=0
  END IF

  IF k11<>0 THEN
    Nline%=Nline%+1
    IF Nu%(n11%)=0 THEN Vn%=n11% ELSE Vn%=Nu%(n11%)
    LSET da2$=MKI$(Vn%)
    LSET db2$=MKI$(Element%)
    LSET dc4$=MKS$(k11)
    LSET dd4$=MKS$(Vc(n11%)-Vc)
    PUT #3
    ' PRINT LOC(2);CVI(da2$);CVI(db2$);CVS(dc4$);CVS(dd4$)
  END IF
  IF MENU(0) THEN Abort
  IF K12<>0 THEN
    Nline%=Nline%+1
    IF Nu%(n12%)=0 THEN Vn%=n12% ELSE Vn%=Nu%(n12%)
    LSET da2$=MKI$(Vn%)
    LSET db2$=MKI$(Element%)
    LSET dc4$=MKS$(K12)
    LSET dd4$=MKS$(Vc(n12%)-Vc)
    PUT #3
```

```
        ' PRINT LOC(2);CVI(da2$);CVI(db2$);CVS(dc4$);CVS(dd4$)
      END IF
      IF MENU(0) THEN Abort
      IF k21<>0 THEN
        Nline%=Nline%+1
        IF Nu%(n21%)=0 THEN Vn%=n21% ELSE Vn%=Nu%(n21%)
        LSET da2$=MKI$(Vn%)
        LSET db2$=MKI$(Element%)
        LSET dc4$=MKS$(k21)
        LSET dd4$=MKS$(Vc(n21%)-Vc)
        PUT #3
        ' PRINT LOC(2);CVI(da2$);CVI(db2$);CVS(dc4$);CVS(dd4$)
      END IF
      IF MENU(0) THEN Abort
      IF k22<>0 THEN
        Nline%=Nline%+1
        IF Nu%(n22%)=0 THEN Vn%=n22% ELSE Vn%=Nu%(n22%)
        LSET da2$=MKI$(Vn%)
        LSET db2$=MKI$(Element%)
        LSET dc4$=MKS$(k22)
        LSET dd4$=MKS$(Vc(n22%)-Vc)
        PUT #3
        ' PRINT LOC(2);CVI(da2$);CVI(db2$);CVS(dc4$);CVS(dd4$)
      END IF
      IF MENU(0) THEN Abort

      sptp&=spt& : spt&=pnt&(spt&)
    WEND
    LSET da2$=MKI$(Node%)
    LSET db2$=MKI$(-1)
    LSET dc4$=MKS$(-c1-c2)
    LSET dd4$=MKS$(0)
    PUT #3, currentnode&
    ' PRINT LOC(2);CVI(da2$);CVI(db2$);CVS(dc4$);CVS(dd4$)
    currentnode&=currentnode&+Nline%+1
  END IF
NEXT Node%
LSET da2$=MKI$(-1)
LSET db2$=MKI$(-1)
LSET dc4$=MKS$(0)
LSET dd4$=MKS$(0)
PUT #3, currentnode&


' Make a note of needed tempory variables
' Trash info in v&(), pnt&(), elem%(), Nu%(), Vc(), Curd()


WINDOW 2,"Compacting Equations"
currentnode&=currentnode&-1&
zero$=MKS$(0)
nsub&=0
FOR i%=1 TO Nodes%
  Vc(i%)=0
NEXT i%
FOR i&=2& TO currentnode&
  GET #3, i&
  IF dd4$<>zero$ THEN
    nsub&=nsub&+1&
    pnt&(nsub&)=i&
    Vn%=CVI(da2$)
    elem%(nsub&)=Vn%
    Vc(Vn%)=-1
  END IF
```

```
  IF MENU(0) THEN Abort
NEXT i&

' Add temporary nodes to dependent equations

isub%=0
i&=1&
WHILE i&<=nsub&
  GET #3, pnt&(i&)
  dnodes%=dnodes%+1
  isub%=isub%+1
  Vtest$=da2$ :  Vctest$=dd4$
  ' Write new dependent equation
  LSET ba2$=MKI$(Nodes%+isub%)
  LSET bb2$=Vtest$
  LSET bc4$=Vctest$
  PUT #1, StartDequ&+dnodes%
  ' Change node number
  LSET da2$=MKI$(Nodes%+isub%)
  LSET dd4$=zero$
  PUT #3, pnt&(i&)
  ' PRINT pnt&(i&);StartDequ&+dnodes%
  pnt&(i&)=StartDequ&+dnodes%
  ' Look for other similar nodes to change
  j&=i&+1
  WHILE j&<=nsub&
    GET #3, pnt&(j&)
    IF da2$=Vtest$ AND dd4$=Vctest$ THEN
      LSET da2$=MKI$(Nodes%+isub%)
      LSET dd4$=zero$
      PUT #3, pnt&(j&)
      nsub&=nsub&-1&
      FOR k&=j& TO nsub&
        pnt&(k&)=pnt&(k&+1&)
        elem%(k&)=elem%(k&+1&)
      NEXT k&
      j&=j&-1&
    END IF
    j&=j&+1&
  WEND
  IF MENU(0) THEN Abort
  i&=i&+1&
WEND
Startlequ&=StartDequ&+dnodes%+1&  ' Start of Independent equations
LSET ba2$=MKI$(dnodes%)
LSET bb2$=MKI$(0) ' Not Used  (Number of extra temporary nodes required in solver)
LSET bc4$=MKL$(Startlequ&)
PUT #1, StartDequ&


' Write independent equations in compact form
' Trash info in v&(), pnt&(), elem%(), Nu%(), Vc(), Curd()


WINDOW 2,"Writing Equation for Node"
GET #1, Startlequ&  ' Start of independent node equations
inodes%=0 ' Number of equations for independent nodes
ilines%=0 ' Number of file lines required for independent nodes
GET #3, 1
n1%=CVI(da2$) :  n2%=CVI(db2$) :  K12=CVS(dc4$)
WHILE n1%>0
  WHILE n2%=-1
    Node%=n1% :  c1=K12
    GET #3
```

```
  n1%=CVI(da2$) : n2%=CVI(db2$) : K12=CVS(dc4$)
WEND
isub%=0
' PRINT "!!!";Node%;c1
WHILE n2%>=0
  isub%=isub%+1
  Nu%(isub%)=n1% : v&(isub%)=n2% : Curd(isub%)=K12
  ' PRINT "@";isub%,n1%;n2%;K12
  GET #3
  n1%=CVI(da2$) : n2%=CVI(db2$) : K12=CVS(dc4$)
WEND

' Sort out equations to eliminate duplication

Nline%=0
FOR i%=1 TO isub%
  n3%=Nu%(i%)
  IF n3%>0 THEN
    Nline%=Nline%+1
    n4%=v&(i%)
    FOR j%=i%+1 TO isub%
      IF n3%=Nu%(j%) AND n4%=INT(v&(j%)) THEN
        Nu%(j%)=0
        Curd(i%)=Curd(i%)+Curd(j%)
      END IF
    NEXT j%
  END IF
  IF MENU(0) THEN Abort
NEXT i%

' Write independent Equation

ilines%=ilines%+1
inodes%=inodes%+1
PRINT inodes%
LSET ba2$=MKI$(Node%)
LSET bb2$=MKI$(Nline%)
LSET bc4$=MKS$(c1)
PUT #1, Startlequ&+ilines%
' PRINT "##";Startlequ&+ilines%,CVI(ba2$);CVI(bb2$);CVS(bc4$)
FOR i%=1 TO isub%
  n3%=Nu%(i%)
  IF n3%>0 THEN
    ilines%=ilines%+1
    LSET ba2$=MKI$(n3%)
    LSET bb2$=MKI$(INT(v&(i%)))
    LSET bc4$=MKS$(Curd(i%))
    PUT #1, Startlequ&+ilines%
    ' PRINT "##";Startlequ&+ilines%,CVI(ba2$);CVI(bb2$);CVS(bc4$)
  END IF
  IF MENU(0) THEN Abort
NEXT i%

' See if temporary equations needs to be writen

IF Vc(Node%) THEN
  FOR i&=1& TO nsub&
    IF elem%(i&)=Node% THEN
      ilines%=ilines%+1
      inodes%=inodes%+1
      GET #1, pnt&(i&)
      Vtest$=bb2$
      LSET bb2$=MKI$(1)
      PUT #1, Startlequ&+ilines%
```

```
            ' PRINT "@@1";StartIequ&+ilines%,CVI(ba2$);CVI(bb2$);CVS(bc4$)
            ilines%=ilines%+1
            writelement: LSET ba2$=Vtest$
            LSET bb2$=MKI$(0)
            LSET bc4$=MKS$(1!)
            PUT #1, StartIequ&+ilines%
            ' PRINT "@@2";StartIequ&+ilines%,CVI(ba2$);CVI(bb2$);CVS(bc4$)
          END IF
        NEXT i&
        IF MENU(0) THEN Abort
      Hagain: END IF

WEND
CLOSE #3
StartBequ&=StartIequ&+ilines%+1& ' Start of Flux density equations
LSET ba2$=MKI$(inodes%)
LSET bb2$=MKI$(0) ' Blank
LSET bc4$=MKL$(StartBequ&)
PUT #1, StartIequ& ' inodes%, blank%, StartBequ&


' Write Permeability equations.


GET #1, StartBequ& ' Start of Permeability equations.
inodes%=0 ' Number of equations for Permeability
FOR j%=1 TO Elements%
  m%=mate%(j%)
  IF m%>2 THEN
    n1%=Ni%(j%,0) : n2%=Ni%(j%,1) : n3%=Ni%(j%,2) : n4%=Ni%(j%,3)
    mpoint: SELECT CASE Type%(j%)
     CASE 1
      n1i%=n1% : n1j%=n2% : n1k%=n4%
      n2i%=n2% : n2j%=n3% : n2k%=n4%
     CASE 2
      n1i%=n1% : n1j%=n2% : n1k%=n3%
      n2i%=n1% : n2j%=n3% : n2k%=n4%
     CASE 0                ' Triangle- Only one will produce nonzero area.
      n1i%=n1% : n1j%=n2% : n1k%=n3%
      n2i%=n2% : n2j%=n3% : n2k%=n4%
     CASE ELSE
      PRINT "Error in element types...." : INPUT x : STOP ' This shouldn't happen !
    END SELECT
    x1i=x(n1i%) : y1i=y(n1i%)
    x1j=x(n1j%) : y1j=y(n1j%)
    x1k=x(n1k%) : y1k=y(n1k%)
    a14=((x1i-x1j)*(y1i-y1k)+(y1i-y1j)*(x1k-x1i)) ' 2*Area  positive direction.
    IF a14<>0 THEN
      b1i=y1j-y1k : c1i=x1k-x1j
      b1j=y1k-y1i : c1j=x1i-x1k
      inodes%=inodes%+1
      LSET aa2$=MKI$(j%)
      LSET ab2$=MKI$(n1i%)
      LSET ac2$=MKI$(n1j%)
      LSET ad2$=MKI$(n1k%)
      PUT #1
      ' PRINT "Bn";j%;TAB(10);n1i%;TAB(20);n1j%;TAB(30);n1k%;TAB(40);LOC(1)
      LSET ca4$=MKS$(c1i/a14)
      LSET cb4$=MKS$(c1j/a14)
      PUT #1
      ' PRINT "Bc";j%;TAB(10);CVS(ca4$);TAB(25);CVS(cb4$);TAB(40);LOC(1)
      LSET ca4$=MKS$(b1i/a14)
      LSET cb4$=MKS$(b1j/a14)
      PUT #1
```

```
      ' PRINT "Bb";j%;TAB(10);CVS(ca4$);TAB(25);CVS(cb4$);TAB(40);LOC(1)
    END IF
    x2i=x(n2i%) :  y2i=y(n2i%)
    x2j=x(n2j%) :  y2j=y(n2j%)
    x2k=x(n2k%) :  y2k=y(n2k%)
    a24=((x2i-x2j)*(y2i-y2k)+(y2i-y2j)*(x2k-x2i)) ' 2*Area  positive direction.
    IF a24<>0 THEN
      b2i=y2j-y2k :  c2i=x2k-x2j
      b2j=y2k-y2i :  c2j=x2i-x2k
      inodes%=inodes%+1
      LSET aa2$=MKI$(j%+Elements%)
      LSET ab2$=MKI$(n2i%)
      LSET ac2$=MKI$(n2j%)
      LSET ad2$=MKI$(n2k%)
      PUT #1
      ' PRINT "Bn";j%+Elements%;TAB(10);n2i%;TAB(20);n2j%;TAB(30),n2k%;TAB(40);LOC(1)
      LSET ca4$=MKS$(c2i/a24)
      LSET cb4$=MKS$(c2j/a24)
      PUT #1
      ' PRINT "Bc";j%+Elements%;TAB(10);CVS(ca4$);TAB(25);CVS(cb4$);TAB(40);LOC(1)
      LSET ca4$=MKS$(b2i/a24)
      LSET cb4$=MKS$(b2j/a24)
      PUT #1
      ' PRINT "Bb";j%+Elements%;TAB(10);CVS(ca4$);TAB(25);CVS(cb4$);TAB(40);LOC(1)
    END IF
    IF MENU(0) THEN Abort
  Idle: END IF
NEXT j%

Endequ&=LOC(1)+1 ' End of Flux density equations
LSET ba2$=MKI$(inodes%)
LSET bb2$=MKI$(0) ' Blank
LSET bc4$=MKL$(Endequ&)
PUT #1, StartBequ&

LSET ca4$=MKL$(StartVequ&)
LSET cb4$=MKL$(StartBequ&)
PUT #1, equpt&

PRINT "Node equations completed",TestE%


Finished:
GET #1, 1
IF TestE%=0 THEN LSET ad2$=MKI$(2) ELSE LSET ad2$=MKI$(3)
PUT #1, 1
CLOSE #3
KILL "FE.scrap"

GoHome:
WINDOW CLOSE 2
WINDOW 1
readnuvc: CHAIN WhoAmI$

Abort:
menu0%=0
CLOSE #3
KILL "FE.scrap"
GOTO GoHome

END
```

EQUATION SOLVER PROGRAM

The equation solver takes the equations stored in the file and solves them in an iterative fashion. The potential of each independent node are calculated based on the current estimated potentials of the surrounding nodes. This process is repeated Nodes4% times and then the permeability of each non-linear element is calculated from BH lookup tables. The information in these tables is for 1010 silicon steel.
There is no test for completion. Instead, the potential of node Pnode% is printed out after every iteration. When the value of this potential stops changing, the solution is complete and the user manually returns to the main program.

```
' Finite Element Solver Program
' 1 Meg Memory Version  9/15/89
' Can compute any problem with
' <2000 nodes,  <2000 square elements, and <32500 equations
' Requires only 1 Meg computer and no scratch files

COMMON WhoAmI$,FileName$,Printf%,menu0%,menu1%
menu0%=0 ' Don't return to Solve command when returning.

CLEAR ,500000&
'CLEAR ,FRE(-1)+FRE(0)-50000&  ' CLEAR Kills COMMON variables !
PRINT FRE(0),FRE(-1),FRE(-2)

WINDOW 2,"Solve Mesh"
DIM v(4000),u(4000)
DIM vn%(4000),nl%(4000),cnst(4000)
Begin: DIM v%(18000),kij(18000),ul%(18000),ck(18000)
DIM ci(4000),cj(4000),bi(4000),bj(4000)
DIM en%(4000),v1%(4000),v2%(4000),v3%(4000)

MENU RESET
MENU 1,0,1,"Status"
MENU 1,1,1,"Return to Main Program"
MENU 1,2,0,"-"
MENU 1,3,1,"Abort without saving results"
MENU 2,0,1,"              "
MENU 3,0,1,"* Solving Equations *"

IF FileName$="" THEN FileName$=FILES$(1,"TEXTMESH") :  OPEN "R",#1,FileName$,8
U0=.000001256637#
FIELD #1, 2 AS aa2$, 2 AS ab2$, 2 AS ac2$, 2 AS ad2$
FIELD #1, 2 AS ba2$, 2 AS bb2$, 4 AS bc4$
FIELD #1, 4 AS ca4$, 4 AS cb4$

GET #1, 1
Nodes%=CVI(aa2$) :  Elements%=CVI(ab2$)
'PRINT Nodes%,Elements%

equpt&=2+3*Nodes%+5*Elements%

GET #1, equpt&
StartVequ&=CVL(ca4$)

GET #1, StartVequ&
Vnode%=CVI(ba2$)
Pnode%=CVI(bb2$)
StartDequ&=CVL(cb4$)

GET #1, StartDequ&
Dnode%=CVI(ba2$)
StartIequ&=CVL(bc4$)
```

```
GET #1, StartIequ&
Indnode%=CVI(ba2$)
StartBequ&=CVL(bc4$)

GET #1, StartBequ&
Bnode%=CVI(ba2$)
Endequ&=CVL(bc4$)
Bnode1%=Bnode%-1

PRINT "Reading in initial Node potentials and permabilities"
vpt&=1+2*Nodes%
v(0)=1
GET #1, vpt&
FOR i%=1 TO Nodes%
   GET #1
   v(i%)=CVS(ca4$)
   IF MENU(0)=1 THEN CHAIN WhoAmI$
NEXT i%
upt&=1+3*Nodes%+2*Elements%
u(0)=1
FOR i%=1 TO Elements%
   GET #1, i%+upt&
   u(i%)=CVS(ca4$)
   u(i%+Elements%)=CVS(cb4$)
   loop: IF MENU(0)=1 THEN CHAIN WhoAmI$
NEXT i%

mselect: PRINT "Reading Set Node Potentials"
GET #1, StartVequ&
'PRINT StartVequ&,Vnode%
while2: FOR i%=1 TO Vnode%
   scase: GET #1
   v(CVI(ba2$))=CVS(bc4$)
   IF MENU(0)=1 THEN CHAIN WhoAmI$
   ' PRINT "V"+MID$(STR$(CVI(ba2$)),2)+"=";CVS(bc4$)
NEXT i%

' Reading dependent nodes
'   This is necessary because if potentials are already defined, then
'   then some other temporary potentials must be defined.

GET #1, StartDequ&
FOR i%=1 TO Dnode%
   GET #1
   v(CVI(ba2$))=v(CVI(bb2$))+CVS(bc4$)
   IF MENU(1)=3 THEN CHAIN WhoAmI$
NEXT i%

PRINT "Number Crunching"
Nodes4%=SQR(Nodes%)/3+.5

npos%=-1
ipos&=-1
GET #1, StartIequ&
FOR i%=1 TO Indnode%
   GET #1
   npos%=npos%+1
   Vnum%=CVI(ba2$)
   nline%=CVI(bb2$)
   Finished: Num=CVS(bc4$)
   vn%(npos%)=Vnum%
   nl%(npos%)=nline%
   cnst(npos%)=Num
```

```
    Den=0
    GoHome: FOR j%=1 TO nline%
      GET #1
      ipos&=ipos&+1
      v%=CVI(ba2$)
      u%=CVI(bb2$)
      error1: ck=CVS(bc4$)
      kij=u(u%)*ck
      v%(ipos&)=v%
      u1%(ipos&)=u%
      ck(ipos&)=ck
      kij(ipos&)=kij
      Den=Den+kij
      Num=Num+v(v%)*kij
    NEXT j%
    IF MENU(0)=1 THEN GoHome
    v(Vnum%)=Num/Den
NEXT i%
error2: iequ&=ipos&

GET #1, StartDequ&  ' dependent potentials.
FOR i%=1 TO Dnode%
    GET #1
    npos%=npos%+1
    vn%(npos%)=CVI(ba2$)
    nl%(npos%)=CVI(bb2$)
    cnst(npos%)=CVS(bc4$)
    IF MENU(0)=1 THEN GoHome
NEXT i%

GET #1, StartBequ&
Abort: FOR bpos%=0 TO Bnode1%
    GET #1
    en%(bpos%)=CVI(aa2$)
    v1%(bpos%)=CVI(ab2$)
    v2%(bpos%)=CVI(ac2$)
    v3%(bpos%)=CVI(ad2$)
    GET #1
    ci(bpos%)=CVS(ca4$)
    cj(bpos%)=CVS(cb4$)
    GET #1
    bi(bpos%)=CVS(ca4$)
    bj(bpos%)=CVS(cb4$)
    IF MENU(0)=1 THEN GoHome
NEXT bpos%

nloop:

FOR Eloop%=1 TO Nodes4%  ' Loop throught independent equations this many times before updating u
    npos%=-1
    ipos&=-1
    FOR i%=1 TO Indnode%
      npos%=npos%+1
      Vnum%=vn%(npos%)
      nline%=nl%(npos%)
      Num=cnst(npos%)
      Den=0
      FOR j%=1 TO nline%
        ipos&=ipos&+1
        kij=kij(ipos&)
        Den=Den+kij
        Num=Num+v(v%(ipos&))*kij
      NEXT j%
      IF MENU(0)=1 THEN GoHome
```

```
      v(Vnum%)=Num/Den
    NEXT i%
    PRINT v(Pnode%)
NEXT Eloop%
' PRINT v(Pnode%);"   Time =";TIMER-xtm&

' B equations may require these dependent potentials.
FOR i%=1 TO Dnode%
    npos%=npos%+1
    v(vn%(npos%))=v(nl%(npos%))+cnst(npos%)
    IF MENU(0)=1 THEN GoHome
NEXT i%

' Given B^2 find u=1/μ

FOR bpos%=0 TO Bnode1%
    e%=en%(bpos%)
    u=u(e%)
    v3=v(v3%(bpos%))
    v13=v(v1%(bpos%))-v3
    v23=v(v2%(bpos%))-v3
    bx=v13*ci(bpos%)+v23*cj(bpos%)
    by=v13*bi(bpos%)+v23*bj(bpos%)
    FINDUB (bx*bx+by*by)*u,u(e%)
    IF MENU(0)=1 THEN GoHome
NEXT bpos%

FOR i&=0 TO iequ&
    kij(i&)=u(u1%(i&))*ck(i&)
NEXT i&

GOTO nloop


GoHome: ' STOP
PRINT "Exiting early,  saving results so far..."
IF MENU(1)=3 THEN CHAIN WhoAmI$  ' This person really wants to leave !

Finished:

' Writing dependent nodes
GET #1, StartDequ&
FOR i%=1 TO Dnode%
    GET #1
    v(CVI(ba2$))=v(CVI(bb2$))+CVS(bc4$)
    IF MENU(1)=3 THEN CHAIN WhoAmI$  ' This person really wants to leave !
NEXT i%

vpt&=1+2*Nodes%
upt&=1+3*Nodes%+2*Elements%
FOR i%=1 TO Nodes%
    LSET ca4$=MKS$(v(i%))
    PUT #1, i%+vpt&
    IF MENU(1)=3 THEN CHAIN WhoAmI$  ' This person really wants to leave !
NEXT i%

FOR i%=1 TO Elements%
    LSET ca4$=MKS$(u(i%))
    LSET cb4$=MKS$(u(i%+Elements%))
    PUT #1, i%+upt&
    IF MENU(1)=3 THEN CHAIN WhoAmI$  ' This person really wants to leave !
NEXT i%
MENU RESET
CHAIN WhoAmI$
```

```
' Given  H*μ0*B=B*B/μr  Find  H*μ0/B=1/μr

SUB FINDUB(x,Y) STATIC
 IF x< .143337936992808# THEN
   IF x< .0132912500793557# THEN
     IF x< 1.50075074867193D-03 THEN
       IF x< 4.43519495082491D-04 THEN
         IF x< 1.98247562498695D-04 THEN
           M#=0 : c#=.00025
         ELSE
           M#= .2364707969374813# : c#= 2.03120240903375D-04
         END IF
       ELSE
         IF x< 8.819998755486231D-04 THEN
           M#= .3238463873585482# : c#= 1.64367463157289D-04
         ELSE
           M#= .350707216206927# : c#= 1.406762154558863D-04
         END IF
       END IF
     ELSE
       IF x< .0036096004864159# THEN
         IF x< 2.24626054996972D-03 THEN
           M#= .3594421396492 98# : c#= 1.27567272560156D-04
         ELSE
           M#= .3484327029936 48# : c#= 1.52297335797132D-04
         END IF
       ELSE
         IF x< 5.93505072258168D-03 THEN
           M#= .331118707005512# : c#= 2.147939441377122D-04
         ELSE
           IF x< 9.363599716406721D-03 THEN
             M#= .3091685836687 35# : c#= 3.45069039508408D-04
           ELSE
             M#= .2800656938351602# : c#= 6.175768505006809D-04
           END IF
         END IF
       END IF
     END IF
   ELSE
     IF x< .05319999891622 07# THEN
       IF x< .02460777412196 78# THEN
         IF x< .0183708005745 81# THEN
           M#= .2618342218820 22# : c#= 8.598959035446059D-04
         ELSE
           M#= .2437078744305423# : c#= 1.192891417721294D-03
         END IF
       ELSE
         IF x< .0324900008730129# THEN
           M#= .2296306051001 22# : c#= 1.539301681658383D-03
         ELSE
           IF x< .0422077506956306# THEN
             M#= .2160994036109 92# : c#= 1.97893042985315D-03
           ELSE
             M#= .2001409995449 27# : c#= 2.652498770173752D-03
           END IF
         END IF
       END IF
     ELSE
       IF x< .0890819988658082# THEN
         IF x< 6.892100017067312D-02 THEN
           M#= .1971884780989 47# : c#= 2.809572907899 99D-03
         ELSE
```

```
            M#= .1884827115790814# :  c#= 3.40958304370152D-03
          END IF
        ELSE
          IF x< .1020529121462594# THEN
            M#= .1850257624991454# :  c#= 3.71753497771955 2D-03
          ELSE
            IF x< .1183359990843981# THEN
              M#= .1842431321826794# :  c#= 3.79740468064885D-03
            ELSE
              M#= .1879854316924574# :  c#= 3.35455592928614D-03
            END IF
          END IF
        END IF
      END IF
    END IF
  END IF
ELSE
  IF x< 2.030339998949464# THEN
    IF x< .4995462478981502# THEN
      IF x< .2812470752111964# THEN
        IF x< .1824680002758821# THEN
          M#= .1891129107224 3# :  c#= 3.192945411127214D-03
        ELSE
          M#= .1807566812524 57# :  c#= 4.717689892359634D-03
        END IF
      ELSE
        IF x< .3465098408420611# THEN
          M#= .1701129239472231# :  c#= .0077112155037144#
        ELSE
          IF x< .4093783030777 75# THEN
            M#= .1632181542895672# :  c#= 1.010032104043142D-02
          ELSE
            M#= .1552872229413 54# :  c#= 1.334707225758943D-02
          END IF
        END IF
      END IF
    ELSE
      IF x< .8862614299925 11# THEN
        IF x< .6398822406740 94# THEN
          M#= .1437296630767 72# :  c#= .01912060792279 95#
        ELSE
          M#= .1288163789668 54# :  c#= 2.866335357486321D-02
        END IF
      ELSE
        IF x< 1.094710355680 26# THEN
          M#= .1142744694520 66# :  c#= 4.155128709626112D-02
        ELSE
          IF x< 1.426631848172 63# THEN
            M#= .1003880216307 04# :  c#= 5.675292532991951D-02
          ELSE
            M#= .0828135878631 228# :  c#= .0818251722563 51#
          END IF
        END IF
      END IF
    END IF
  ELSE
    IF x< 20.34594676792601# THEN
      IF x< 5.083272842696882# THEN
        IF x< 3.428283000700 5# THEN
          M#= .0596364952974 854# :  c#= .1288825503517 19#
        ELSE
          M#= 4.039802642433373D-02 :  c#= .1948374661490501#
        END IF
      ELSE
        IF x< 9.149770479696492# THEN
```

```
            M#= 2.453234488638381D-02 :  c#= .2754870542417882#
           ELSE
            IF x< 12.8705572079986# THEN
              M#= .0149570400771303# :  c#= .363098895519592#
             ELSE
              M#= .0092998883903412# :  c#= .4359095899938737#
            END IF
          END IF
        END IF
       ELSE
        IF x< 40.06243523531 94# THEN
          IF x< 27.46593297750 11# THEN
            M#= 5.823974729740 31D-03 :  c#= .5066303442472294#
           ELSE
            M#= 3.788640135875 88D-03 :  c#= .5625327077890993#
          END IF
         ELSE
          IF x< 121.40385730177 13# THEN
            M#= 1.345372885237 43D-03 :  c#= .660415943780379#
           ELSE
            IF x< 9999.991009271052# THEN
              M#= 1.784158975957 96D-05 :  c#= .821583363740207#
             ELSE
              M#=0 :  c#=1
            END IF
          END IF
        END IF
      END IF
    END IF
  END IF
  Y=x*M#+c#
END SUB
```

INITIAL DISTRIBUTION

Copies

3 ONT

2	211	Remmers
1	211	Gagorik

5 NAVSEA

1	05Z	Graham
1	08K	Levin
1	56Z	Yee
1	5112	Tobin
1	92R	Sofia

12 DTIC

CENTER DISTRIBUTION

Copies	Code	Name
1	0113	Winegrad
1	0114	Becker
1	1413	Drake
1	2702	Levedahl
1	2704	Quandt
1	271	Stevens
1	2711	Bagley
5	2711	Cannell
5	2711	McConnell
15	2711	Smith
1	2712	Superczynski
1	27B	Robey
1	27BA	Cox
1	27B	Dunnington
1	522.1	TIC (C)
1	522.2	TIC (A)
1	5231	Office Services

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	JUNE 1992	JUNE 1988 - JUNE 1992 CARDEROCKDIV-PAS-92-27

4. TITLE AND SUBTITLE

MAGNETIC FLUX - LOAD CURRENT INTERACTIONS IN FERROUS CONDUCTORS

5. FUNDING NUMBERS

WORK UNIT # 1-2711-100

PROGRAM ELEMENT # 62121N
PROJECTS # RH21E41
" # RH21E42

6. AUTHOR(S)

MICHAEL J. CANNELL. RICHARD A. McCONNELL

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

ANNAPOLIS DETACHMENT, CARDEROCK DIVISION,
NAVAL SURFACE WARFARE CENTER
ANNAPOLIS, MD 21402-5067

8. PERFORMING ORGANIZATION REPORT NUMBER

CODE 2711
CARDEROCKDIV-PAS-92-27

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

OFFICE OF NAVAL TECHNOLOGY
ONT-211
800 N. QUINCY STREET
ARLINGTON, VA 22217-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED

12b. DISTRIBUTION CODE

STATEMENT A

13. ABSTRACT (*Maximum 200 words*)

A MODELING TECHNIQUE HAS BEEN DEVELOPED TO ACCOUNT FOR INTERACTIONS BETWEEN LOAD CURRENT AND MAGNETIC FLUX IN AN IRON CONDUCTOR. SUCH A CONDUCTOR WOULD BE USED IN THE ACTIVE REGION OF A NORMALLY CONDUCTING HOMOPOLAR MACHINE. THIS APPROACH HAS BEEN EXPERIMENTALLY VERIFIED AND IS APPLICATION TO A REAL MACHINE DEMONSTRATED. ADDITIONALLY, MEASUREMENTS OF THE RESISTIVITY OF STEEL UNDER THE COMBINED EFFECTS OF MAGNETIC FIELD AND CURRENT HAVE BEEN CONDUCTED.

14. SUBJECT TERMS

CONTRA-ROTATING, HOMOPOLAR, MOTOR ELECTRIC, IRON CONDUCTORS, MAGNETORESISTIVE

15. NUMBER OF PAGES

99

16. PRICE CODE

17. SECURITY OF CLASSIFICATION OF REPORT	18. SECURITY OF CLASSIFICATION OF THIS PAGE	19. SECURITY OF CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNLIMITED